

AUTOCAD®

AutoLISP 10.0
Manuale del programmatore

Pubblicazione TD111-005.II

Febbraio 1989

Questo manuale è stato pubblicato nell'ottobre 1987 e si basa sulla Versione 9 di AutoLISP®. E' stato redatto utilizzando il sistema di composizione testi Laserscript® della Command Technology Corp., Oakland, CA. e stampato con la stampante Laserjet Series II della Hewlett-Packard.

AutoLISP è basato su XLISP, un programma elaborato da David Betz. Vogliamo con ciò esprimere la nostra gratitudine a David Betz per il contributo dato col suo programma alla creazione di AutoLISP. Informazioni ulteriori relative a XLISP possono essere ottenute da David Betz, 127 Taylor Road, Peterborough, NH 03458.

Indice

Capitolo 1 Introduzione 1

1.1 Perchè LISP? 2
1.2 Tipi di dati in AutoLISP 2
1.3 Il programma di valutazione di AutoLISP 3
1.4 Convenzioni di AutoLISP 4
1.5 Convenzioni di questo manuale 5
1.6 Gestione degli errori 6

Capitolo 2 Installazione di AutoLISP 7

2.1 Formato di release 7
2.2 AutoLISP normale 7
2.2.1 Configurazione 7
2.2.2 Variabili d'ambiente 7

2.3 AutoLISP Esteso 8
2.3.1 Configurazione 8
2.3.2 Variabili d'ambiente 8
2.3.3 Come servirsi di AutoLISP Esteso 9

2.4 Funzioni automatiche - File "acad.lsp" 9

Capitolo 3 Esempio di programmazione 11

3.1 Considerazioni preliminari 11
3.2 Obiettivo 11
3.3 Convertire gradi in radianti 12
3.4 Ottenere dati relativi al viale 13
3.5 Disegnare il contorno del viale 16
3.6 Disegnare le piastrelle 17
3.7 Aggiungere il comando ad AutoCAD 20
3.8 Eliminare la ripetizione dei comandi 22
3.9 Riassunto 24

Capitolo 4 Funzioni di AutoLISP 25

4.1 La variabile di sistema FLATLAND - Compatibilità con versioni precedenti 25
4.2 (+ <numero> <numero>...) 25
4.3 (- <numero> <numero>...) 25
4.4 (* <numero> <numero>...) 26
4.5 (/ <numero> <numero>...) 26
4.6 (= <atomo> <atomo>...) 26
4.7 (/= <atom1> <atom2>) 27
4.8 (< <atomo> <atomo>...) 27
4.9 (<= <atomo> <atomo>...) 27
4.10 (> <atomo> <atomo>...) 27
4.11 (>= <atomo> <atomo>...) 28
4.12 (~ <numero>) 28
4.13 (1+ <numero>) 28

4.14 (1- <numero>) 28
4.15 (abs <numero>) 28
4.16 (and <espressione>...) 29
4.17 (angle <pt1> <pt2>) 29
4.18 (angtos <angolo> [<modo> [<precisione>]]) 29
4.19 (append <espressione>...) 30
4.20 (apply <funzione> <lista>) 30
4.21 (ascii <stringa>) 30
4.22 (assoc <elemento> <alista>) 31
4.23 (atan <num1> [<num2>]) 31
4.24 (atof <stringa>) 31
4.25 (atoi <stringa>) 32
4.26 (atom <elemento>) 32
4.27 (Boole <funz> <int1> <int2>...) 32
4.28 (boundp <atomo>) 33
4.29 caar, cadr, cddr, cadar, ecc. 33
4.30 (car <lista>) 34
4.31 (cdr <lista>) 34
4.32 (chr <numero>) 35
4.33 (close <descrfile>) 35
4.34 (command <argom>...) 35
Pausa per immissione dati da parte dell'utente 36
4.35 (cond (<test1> <risultato1>) ...) 37
4.36 (cons <nuovo primo elemento> <lista>) 37
4.37 (cos <angolo>) 38
4.38 (defun <sim> <lista argomenti> <espr>...) 38
4.38.1 Librerie di funzioni e caricamento automatico 39
4.38.2 Funzioni C:XXX -- Aggiungere comandi AutoCAD 39
4.38.3 Funzioni S::XXX - Esecuzione automatica 40

4.39 (distance <pt1> <pt2>) 40
4.40 (eq <espr1> <espr2>) 41
4.41 (equal <espr1> <espr2>) <approssimazione> 41
4.42 (eval <espr>) 42
4.43 (exp <numero>) 42
4.44 (expt <base> <esponente>) 42
4.45 (findfile <nome file>) 42
4.46 (fix <numero>) 43
4.47 (float <numero>) 43
4.48 (foreach <nome> <lista> <espr>...) 43
4.49 (gcd <num1> <num2>) 44
4.50 (getangle [<pt>] [<richiesta>]) 44
4.52 (getdist [<pt>] [<richiesta>]) 45
4.53 (getenv <nome della variabile>) 45
4.54 (getint [<richiesta>]) 46
4.57 (getpoint [<pt>] [<richiesta>]) 47
4.58 (getreal [<richiesta>]) 48
4.59 (getstring [<cr>] [<richiesta>]) 48
4.60 (getvar <nome variabile>) 48
4.61 (graphscr) 48
4.62 (if <espr test> <espr then> [<espr else>]) 49
4.64 (inters <pt1> <pt2> <pt3> <pt4> [<sulseg>] 51
4.65 (itoa <int>) 52
4.66 (lambda <argomenti> <espr>...) 52

4.67 (last <lista>) 52
4.68 (length <lista>) 53
4.69 (list <espr>...) 53
4.70 (listp <elemento>) 53
4.71 (load <nome file>) [<per difetto>]) 53
4.72 (log <numero>) 54
4.73 (logand <numero> <numero>...) 54
4.74 (logior <numero> <numero>...) 54
4.75 (lsh <num1> <num bit>) 55
4.76 (mapcar <funzione> <lista1> ... <listan>) 55
4.77 (max <numero> <numero>...) 56
4.78 (member <espr> <lista>) 56
4.79 (menucmd <stringa>) 56
4.80 (min <numero> <numero>...) 57
4.81 (minusp <elemento>) 57
4.82 (not <elemento>) 57
4.83 (nth <n> <lista>) 58
4.84 (null <elemento>) 58
4.85 (numberp <elemento>) 58
4.86 (open <nome file> <modo>) 59
4.87 (or <espr>...) 60
4.88 (osnap <pt> <stringa modo>) 60
4.89 pi 60
4.90 (polar <pt> <angolo> <distanza>) 60
4.91 (prin1 <espr> [<desc file>]) 60
4.92 (princ <espr> [<desc file>]) 62
4.93 (print <espr> [<desc file>]) 62
4.94 (progn <espr>...) 62
4.95 (prompt <mess>) 62
4.96 (quote <espr>) 62
4.97 (read <stringa>) 63
4.98 (read-char [<desc file>]) 63
4.99 (read-line [<desc file>]) 64
4.100 (redraw [<nome entità> [<modo>]]) 64
4.101 (rem <num1> <num2>...) 65
4.102 (repeat <numero> <espr>...) 65
4.103 (reverse <lista>) 65
4.104 (rtos <numero> [<modo> [<precisione>]]) 65
4.105 (set <simb> <espr>) 66
4.106 (setq <simb1> <espr1> [<simb2> <espr2>]...) 66
4.107 (setvar <nome var> <valore>) 67
4.108 (sin <angolo>) 67
4.109 (sqrt <numero>) 68
4.110 (strcase <stringa> [<espr>]) 68
4.111 (strcat <stringa1> <stringa2>...) 68
4.112 (strlen <stringa>) 68
4.113 (subst <nuovo elem> <elem prec> <lista>) 68
4.114 (substr <stringa> <inizio> [<lung>]) 69
4.115 (terpri) 69
4.116 (textscr) 69
4.117 (trace <funzione>) 70
4.118 (trans <pt> <da> <a> [<spostamento>] 70
4.119 (type <elemento>) 72
4.120 (untrace <funzione>...) 73

4.121 (ver) 73
4.122 (vports) 73
4.123 (while <espr test> <espr>...) 74
4.124 (write-char <num> [<desc file>]) 74
4.125 (write-line <stringa> [<desc file>]) 75
4.126 (zerop <elemento>) 75
4.127 (*error* <stringa>) 75

Capitolo 5 Accesso a entità e dispositivi 77

5.1 Tipi di dati speciali 77
5.2 Funzioni relative al gruppo di selezione 77
5.2.1 (ssget [<modo>] [<pt1> [<pt2>]]) 77
Filtri SSGET (ssget "X") 78
5.2.2 (sslength <gs>) 79
5.2.3 (ssname <gs> <indice>) 79
5.2.4 (ssadd [<nome ent> [<gs>]]) 80
5.2.5 (ssdel <nome ent> <gs>) 80
5.2.6 (ssmemb <nome ent> <gs>) 81

5.3 Funzioni relative ai nomi d'entità 81
5.3.1 (entnext [<nome ent>]) 81
5.3.2 (entlast) 81
5.3.3 (entsel [<messaggio>]) 82
5.3.4 (handent <identificatore>) 82

5.4 Funzioni relative ai dati di entità 83
5.4.1 (entdel <nome ent>) 83
5.4.2 (entget <nome ent>) 83
5.4.3 (entmod <lista ent>) 86
5.4.4 (entupd <nome ent>) 87
5.4.5 Restrizioni 87

5.5 Uso di nomi d'entità e di gruppi di selezione in AutoCAD 88
5.6 Note sulle curve del tipo V e sulle splinee 88
5.7 Accesso alla tavola dei simboli 88
5.7.1 (tblnext < nome della tavola> [<primo>]) 88
5.7.2 (tblsearch <nome della tavola> <simbolo>)[<prossimo>] 90

5.8 Accesso allo schermo grafico e ai dispositivi di input 90
5.8.1 (grclear) 91
5.8.2 (grdraw <da> <a> <colore> [<evidenziare>]) 91
5.8.3 (grtext [<casella> <testo> [<evidenziare>]]) 91
5.8.4 (grread [<aggiornare>]) 92

Capitolo 6 Gestione della memoria 95

6.1 Adattare la memoria alle esigenze di AutoLISP 95
6.2 Ricupero di memoria nodale 97
6.3 Paginazione virtuale di funzioni 98
6.4 Considerazioni tecniche 98
6.4.1 Spazio nodale 99
6.4.2 Spazio di stringa 99
6.4.3 Archiviazione di simboli 99

6.4.4 Allocazione manuale 100
6.4.5 Statistiche di memoria 101
6.4.6 Paginazione virtuale di funzioni 101

Appendice A Programmi di AutoLISP 103

A.1 Come caricare i programmi 103
A.2 Come lanciare i programmi 104
A.3 Programmi applicativi 104
A.3.1 3D - Costruzione di oggetti tridimensionali 104
A.3.1.1 SCATOLA - Scatola o cubo 3D 106
A.3.1.2 CONO 106
A.3.1.3 CUPOLA o SCODELLA - Rete poligonale emisferica 107
A.3.1.4 RETE - Rete planare semplice 108
A.3.1.5 PIRAMIDE 108
A.3.1.6 SFERA 109
A.3.1.7 TORO 110
A.3.1.8 CUNEO 111

A.3.2 3DARRAY - Serie 3D rettangolari e polari 112
A.3.3 AFKINET, AFLIX, AFWALK - File di AutoFlix aggiornati 113
A.3.4 ASCTEXT - Inserimento di un testo proveniente da un file ASCII 114
A.3.5 ASHADE - File "ashade.lsp" aggiornato 114
A.3.6 ATTREDEF - Aggiornamento e ridefinizione di attributi 115
A.3.7 CHGTEXT - Ricerca e sostituzione di un testo 115
A.3.8 DELLAYER - Cancellazione di tutte le entità su un piano 115
A.3.9 EDGE - Modifica la visibilità di spigoli di facce 3D 115
A.3.10 LEXPLODE - Nuova versione del comando ESPLOSO 116
A.3.11 REF - Ottenimento di un punto relativo 116
A.3.12 SETUP - Definizione della scala e dei limiti di un disegno 116
A.3.13 SSX - Versione semplificata di (ssget "x") 117

A.4 Esempi di programmazione 117
A.4.1 AXROT - Rotazione di entità intorno ad un asse 118
A.4.2 CHFACE - Spostamento dei vertici di una faccia 3D 118
A.4.3 CL - Costruzione di linee centrali 118
A.4.4 DRAWMAN - Esempio di gestione di entità 118
A.4.5 FACT - Calcolo fattoriale 119
A.4.6 FCOPY - Copia di un file di testo in un'altro 119
A.4.7 FPLOT - Funzione di riproduzione grafica di due variabili 120
A.4.8 FPRINT - Lista file di testo sullo schermo 121
A.4.9 PROJECT - Proiezione di un modello 3D sull'UCS corrente 122
A.4.10 RPOLY - Modifiche ad un poligono 122
A.4.11 SLOT - Costruzione di scanalature e cavità 122
A.4.12 SPIRAL - Costruzione di spirali 2D 123
A.4.13 SQR - Calcolo della radice quadrata 123
A.4.14 TABLES - Visualizzazione/riordino di tavole di simboli 123

Appendice B Messaggi di errore 125

B.1 Errori nei programmi utente 125
B.2 Errori interni 130

Indice analitico i

Capitolo 1

Introduzione

AutoLISP[a] è un'implementazione del linguaggio di programmazione LISP integrata nel pacchetto ADE-3 di AutoCAD[R]. AutoLISP permette agli utenti e ai programmatori di AutoCAD di scrivere macroprogrammi e funzioni in un linguaggio potente e perfettamente idoneo alle applicazioni grafiche. LISP è molto flessibile, facile da imparare e da utilizzare.

AVVISO

Per lavorare efficientemente con AutoCAD non è indispensabile imparare ad usare AutoLISP: se non avete esperienza nella programmazione, vi consigliamo di leggere solo le indicazioni riportate nel Capitolo 2 concernenti l'installazione di AutoLISP, poiché vi permetteranno di sfruttare numerosi menù e applicazioni di AutoCAD che richiedono AutoLISP.

Se invece vi piace programmare, la lettura di questo manuale vi sarà d'aiuto nell'usare AutoLISP per trasformare il programma "general purpose" AutoCAD in un programma ancora più potente in grado di assistervi nel vostro campo specifico.

Questo fascicolo è un manuale di consultazione, non è una presentazione di LISP e quindi non si presta allo studio di questo linguaggio di programmazione. Per acquisire le conoscenze fondamentali richieste dalla programmazione in LISP, consigliamo la lettura di testi introduttivi e didattici dei quali vi proponiamo tre libri in italiano e tre in inglese, rispettivamente:

LISP - Linguaggio e metodologia di programmazione di G. Gini, M. Gini e G. Guida, Edizione clup, Milano 1981

Introduzione al Lisp di Ken Tracton, tradotto dall'inglese, Padova 1984

LISP - Il linguaggio dell'intelligenza artificiale di A.A. Berk, tradotto dall'inglese, Milano 1985

I libri in inglese consigliati sono:

LISP - A gentle introduction to symbolic computation, di David S. Touretzky, Ed. Harper & Row, New York 1984

LISP di Winston e Horn (seconda edizione) e *Looking at LISP* di T. Hasemer, entrambi pubblicati da Addison Weseley.

Il LISP è un linguaggio con diversi dialetti, fra i quali MacLISP, InterLISP, ZetaLISP e Common LISP. La sintassi e le convenzioni adottate da AutoLISP si avvicinano a quelle del Common LISP, ma AutoLISP possiede diverse funzioni addizionali concepite appositamente per AutoCAD. Questo manuale presenta tutte le funzioni di AutoLISP e ne spiega il funzionamento e l'impiego.

1.1 Perchè LISP?

Abbiamo scelto LISP come principale linguaggio di interfaccia per numerose ragioni:

- LISP è particolamente efficiente nella gestione di oggetti e :i in gruppi di diverse dimensioni, il tipo di informazioni cioè con il qu nte lavorano i sistemi di CAD.

- Un interprete di LISP si adatta perfettamente all'ir :trutturata che caratterizza i procedimenti di design.

- LISP è tra i linguaggi di programmazione più semplici d:

- LISP è il linguaggio scelto per la ricerca e lo svilu dell'intelligenza artificiale e dei sistemi esperto.

- Data la semplicità della sintassi in LISP, un interprete amente semplice da implementare e di dimensioni limitate.

In futuro abbiamo intenzione di aggiungere al linguaggio LIS. e applicative. La Autodesk intende comunque lavorare a lunga scadenza con nuovi linguaggi implementati saranno introdotti come alternative in aggiunta ad

1.2 Tipi di dati in AutoLISP

AutoLISP fa uso dei seguenti tipi di dati:

- liste
- simboli
- stringhe
- numeri reali
- numeri interi
- descrittori di file
- "nomi" di entità AutoCAD
- gruppi di selezione AutoCAD
- subrs (funzioni predefinite)

Sui sistemi PC-DOS/MS-DOS, gli interi sono numeri a :ti di segno, quindi compresi tra -32768 e +32767. Sulle stazioni di lavoro a 3: di AutoLISP saranno numeri a 32 bit provvisti di segno con valori oscillanti da + 2.147.483.647. Gli interi trasferibili tra AutoLISP e AutoCAD sono limitati a

I numeri reali sono memorizzati in doppia precisione e v: forniscono almeno 14 cifre significative di precisione. Le stringhe possono avere qualsiasi e la memoria che richiedono viene loro assegnata dinamicamente. L stringe non devono comunque superare i 100 caratteri.

AutoLISP comprende una serie di funzioni incorporat ono una base per la programmazione di applicazioni grafiche bidimensionali e .i. Per lavorare con le coordinate dei punti, occorre ricordare le seguenti conven:

I punti a 2D vengono espessi in forma di liste di due numeri reali (X Y), es.:

```
(3.4 7.52)
```

Il primo valore corrisponde alla coordinata X, il secondo alla Y.

I punti a 3D vengono espressi in forma di liste di tre numeri reali (X Y Z), es.:

```
(3.4 7.52 1.0)
```

Il primo valore corrisponde alla coordinata X, il secondo alla Y e il terzo valore è quello della Z.

Nei casi in cui AutoCAD richiede l'immissione di un certo tipo di dati (ad esempio un punto o un fattore scalare), per fornire il valore desiderato si può impiegare un'espressione AutoLISP di questo tipo oppure una funzione AutoLISP che produca il tipo di valori desiderato.

1.3 Il programma di valutazione di AutoLISP

Alla base di ogni interprete LISP c'è il programma di valutazione. Questo programma esamina ogni riga inserita dall'utente, la valuta e ne restituisce come risultato il valore. Il procedimento di valutazione in AutoLISP opera nel modo seguente:

- I numeri interi, quelli reali, le stringhe, i puntatori di file e le subrs restituiscono il proprio valore come risultato.
- I simboli danno come risultato il valore a loro associato al momento della valutazione.
- Le liste vengono valutate secondo il primo elemento della lista. Se il primo elemento dà come risultato:
  - una lista (o NIL), la lista è interpretata come definizione di funzione e la funzione viene valutata usando i valori degli altri elementi della lista come argomenti.
  - il nome di una funzione predefinita (subr), gli altri elementi della lista vengono considerati come argomenti della subr e sono valutati da questa.

Se si immette un'espressione AutoLISP in risposta a un messaggio di richiesta "Comando:" di AutoCAD, AutoLISP valuta l'espressione e visualizza il risultato ottenuto, dopodiché riappare il messaggio "Comando:". Per la visualizzazione di numeri reali, AutoLISP fornisce fino a 6 cifre significative.

Se è stata digitata o letta da un file un'espressione sbagliata, AutoLISP visualizza:

n>

in cui *n* è un intero che indica il numero di parentesi di chiusura che mancano per completare l'espressione. Quando appare questo messaggio, bisogna fornire il numero di parentesi mancanti per ovviare all'errore. Un errore molto comune è quello di dimenticare le virgolette di chiusura (") di una stringa di testo, nel qual caso le parentesi di chiusura verranno interpretate senza essere valutate e non serviranno quindi a modificare l'espressione. Per correggere questo errore occorre digitare le virgolette seguite da *n* parentesi di chiusura.

1.4 Convenzioni di AutoLISP

Le espressioni di AutoLISP possono essere digitate dalla tastiera mentre si utilizza AutoCAD, possono essere lette a partire da un file ASCII o da un variabile in forma di stringa. In ogni caso bisogna tener conto delle convenzioni seguenti:

- I nomi di simboli possono essere formati da una sequenza qualsiasi di caratteri stampabili ad eccezione di:

  ( ) . ' " ;

- I seguenti caratteri vanno posti alla fine di un nome di simbolo o di una costante numerica.

  ( ) ' " ; (spazio vuoto) (fine di linea)

- Le espressioni possono protrarsi per parecchie righe.

- Molti spazi vuoti tra un simbolo e l'altro equivalgono sempre ad uno spazio solo. Non è obbligatorio rientrare i capoversi, si può comunque farlo per dare più rilievo alla struttura della funzione.

- I nomi dei simboli e delle subrs possono essere indifferentemente in maiuscolo o in minuscolo. I nomi di simboli non possono iniziare con una cifra.

- Le costanti intere possono avere il prefisso opzionale "+" o "-". I numeri interi sono compresi fra -32768 e + 32767.

- Le costanti reali consistono di una o più cifre seguite da un punto decimale, seguito a sua volta da una o più cifre; per esempio ".4" non viene identificato come numero reale. "0.4" invece si. Lo stesso vale, ad esempio, per "5.", che non è considerato come numero reale, mentre "5.0" si. I numeri reali possono anche essere rappresentati in notazione scientifica, nel qual caso contengono una "e" o "E" opzionale seguita dall'esponente del numero.

- Le stringhe di testo sono sequenze di caratteri racchiuse da virgolette. La barra rovesciata (il carattere \), usata all'interno di stringhe, permette di includere caratteri di controllo. I codici riconosciuti sono:

`\\`	corrisponde al carattere "\"
`\e`	corrisponde a "escape" (cambiamento di codice)
`\n`	corrisponde a "newline" (a capo)
`\r`	corrisponde a "return" o "enter"
`\t`	corrisponde a "tab" (tabulatore)
`\nnn`	corrisponde al carattere il cui codice ottale è nnn

- La stringa seguente produrrebbe, ad esempio, un messaggio posto a capo e in principio di riga.

```
(prompt "\nInserisci il primo punto: ")
```

- Il carattere ('), l' apostrofo, può essere usato come abbreviazione della funzione QUOTE. Quindi:

```
'foo
```

equivale a:

```
(quote foo)
```

- Nei programmi AutoLISP caricati da file su disco si possono includere commenti. I commenti sono preceduti da un punto e virgola, e continuano fino al termine della riga. Esempio:

```
; Questa riga è un commento

(setq area (* pi r r))     ; Calcola l'area del cerchio
```

1.5 Convenzioni di questo manuale

Nella descrizione delle funzioni di AutoLISP useremo alcune convenzioni. Esempio:

```
(moo <stringa> <numero>...)
```

Il nome della funzione è mostrato così come deve essere digitato. Gli elementi racchiusi fra le parentesi "<>" che seguono il nome della funzione indicano il numero e il tipo di argomenti richiesti dalla funzione. In questo esempio, la funzione "moo" ha due argomenti richiesti: una stringa e un numero. I puntini stanno ad indicare che possono essere aggiunti altri argomenti numerici. E' importante notare che si devono omettere le parentesi "<>" e i puntini quando si digita l'espressione.

Seguono esempi validi per la funzione "moo" (premesso il formato di richiamo appena descritto):

```
(moo "Ciao" 5)
(moo "Grazie" 1 2 3)
```

Gli esempi che seguono invece, darebbero luogo a condizioni di errore, dato che non sono conformi al formato prescritto:

```
(moo 1 2 3)                  (il primo argomento deve essere una stringa)
(moo "Ciao")                 (uno degli argomenti deve essere numerico)
(moo "eseguire" '(1 2))      (il secondo argomento deve essere un numero, non
                             una lista)
```

Quando un argomento opzionale può figurare una volta ma non può essere ripetuto, esso viene racchiuso fra parentesi quadre ("[]"), come in:

```
(foo <stringa> [<numero>])
```

In questo caso la funzione "foo" richiede un argomento stringa e accetta un argomento numerico opzionale. Ad esempio, i seguenti richiami della funzione "foo" sono corretti:

```
(foo "carica")
(foo "carica" 22)
```

Gli esempi seguenti invece non rispettano il formato prescritto e darebbero luogo ad errori:

```
(foo 44 13)              (il primo argomento deve essere una stringa)
(foo "prego" 44 13)      (troppi argomenti)
```

1.6 Gestione degli errori

Se AutoLISP incontra un errore durante il processo di valutazione, appare un messaggio del tipo:

error: *testo*

dove "*testo*" è il messaggio di errore. Se è definita la funzione *ERROR* (diversa da NIL), AutoLISP, invece di visualizzare il messaggio, esegue questa funzione (con "*testo*" come unico argomento). Se *ERROR* non è definita o ha valore NIL, il processo di valutazione di AutoLISP si arresta e viene visualizzata la traccia della funzione richiamata fino a 100 livelli di profondità.

Capitolo 2

Installazione di AutoLISP

2.1 Formato di release

AutoLISP è contenuto in ogni copia di AutoCAD che include ADE-3. Sotto PC-DOS/MS-DOS il file di sovrapposizione per il programma AutoLISP si chiama "acadl.ovl" ed è contenuto in uno dei dischetti programma di AutoCAD. Sotto altri sistemi il file si chiama "acadl".

Uno dei dischetti programma di AutoCAD contiene il file "readme.doc" che riporta le ultime modifiche ed eventuali aggiornamenti aggiunti in AutoCAD e AutoLISP; assicuratevi di aver letto questo file prima di iniziare a lavorare.

> *Su sistemi PC-DOS/MS-DOS viene fornita una versione addizionale di AutoLISP, chiamata AutoLISP Esteso, consistente dei file "acadlx.ovl", "extlisp.exe" e "remlisp.exe". Rimandiamo ai capitoletti seguenti per ulteriori informazioni.*

2.2 AutoLISP normale

I punti seguenti riguardano solo i sistemi che non utilizzano AutoLISP Esteso. Per lanciare AutoLISP Versione 10 bisogna avere a disposizione:

- Un computer gestito da AutoCAD con una memoria di almeno 640 Kbytes e un disco rigido.
- Una versione compatibile di AutoCAD comprendente ADE-3.

2.2.1 Configurazione

Quando si configura AutoCAD, uno dei parametri operativi da definire riguarda l'utilizzo o meno di AutoLISP. Sotto PC-DOS/MS-DOS, il Configuratore presenta un ulteriore messaggio che domanda se si vuole utilizzare AutoLISP Esteso. Per utilizzare solo AutoLISP normale, occorre rispondere "no". Rimandiamo alla *AutoCAD Installation and Performance Guide* per ulteriori dettagli.

2.2.2 Variabili d'ambiente

Sotto PC-DOS/MS-DOS, una certa quantità della memoria del computer deve essere riservata ad AutoLISP. Se utilizzate un pacchetto applicativo come AutoCAD AEC, consultate il relativo manuale per verificare l'impostazione da dare alle variabili d'ambiente LISPHEAP e LISPSTACK. Rimandiamo alla *AutoCAD Installation and Performance Guide* e al Capitolo 6 di questa guida per ulteriori dettagli.

2.3 AutoLISP Esteso

Per lavorare con AutoLISP Esteso Release 10 bisogna avere a disposizione:

- un computer basato su microprocessore Intel 80286 o 80386, gestito da AutoCAD, con una memoria di almeno 640 Kbytes e un disco rigido
- almeno 512 Kbytes di memoria estesa del tipo IBM AT (e *non* Lotus/Intel/Microsoft) non destinata ad altro impiego (come dischi di RAM).
- PC-DOS/MS-DOS Versione 2.0 o seguenti
- una versione compatibile di AutoCAD comprendente ADE-3.

Se disponete di quanto elencato, potete utilizzare AutoLISP Esteso al posto del AutoLISP normale. AutoLISP Esteso opera in modo 80286/80386 protetto e risiede nella memoria estesa, permettendo così ai programmi di AutoLISP di essere di dimensioni maggiori e gestire una quantità superiore di dati che non AutoLISP normale. L'utilizzo di AutoLISP Esteso al posto dell'AutoLISP normale libera una quantità maggiore di memoria di DOS (al di sotto di 640K) per effettuare paginazioni I/O. AutoLISP Esteso non può funzionare su computer del tipo PC/XT dal momento che la loro unità centrale non gestisce il modo operativo protetto.

Prescindendo dalla maniera di utilizzare la memoria, non esiste nessuna differenza per quanto riguarda le funzioni e le applicazioni tra AutoLISP normale e AutoLISP Esteso

2.3.1 Configurazione

Quando si configura AutoCAD, uno dei parametri operativi da definire riguarda l'utilizzo o meno di AutoLISP. Sotto PC-DOS/MS-DOS, il Configuratore presenta un ulteriore messaggio che domanda se si vuole utilizzare AutoLISP Esteso. Per utilizzare solo AutoLISP Esteso, occorre rispondere "si". Rimandiamo alla *AutoCAD Installation and Performance Guide* per ulteriori dettagli.

2.3.2 Variabili d'ambiente

La variabile d'ambiente LISPXMEM permette di controllare l'area di memoria estesa che AutoLISP Esteso utilizzerà. (Se LISPXMEM non è definita, AutoLISP Esteso utilizzerà tutta la memoria estesa disponibile oltre ai dischi di RAM). AutoLISP Esteso informa AutoCAD dell'area di memoria che sta usando, rendendo quest'area inutilizzabile per paginazione I/O.

Una porzione della memoria estesa assegnata ad AutoLISP Esteso deve essere tenuta da parte come area di stack. Se utilizzate un pacchetto applicativo come AutoCAD AEC, consultate il relativo manuale per verificare l'impostazione da dare alla variabile d'ambiente LISPSTACK. (AutoLISP Esteso non si serve della variabile d'ambiente LISPHEAP)

Rimandiamo alla *AutoCAD Installation and Performance Guide* e al Capitolo 6 di questa guida per ulteriori dettagli.

2.3.3 Come servirsi di AutoLISP Esteso

AutoLISP Esteso è implementato come programma separato, EXTLISP, che deve essere lanciato prima di AutoCAD. EXTLISP è un programma residente del tipo "terminate-and-stay" (TSR) che risiede parzialmente nella memoria estesa e comunica con AutoCAD tramite il file di sovrapposizione "acadlx.ovl". Potete aggiungere il comando EXTLISP necessario nel vostro "autoexec.bat", se lo desiderate.

Potete rimuovere EXTLISP dalla memoria dopo aver lanciato AutoCAD. Per fare questo, richiamate il comando REMLISP mentre siete in DOS. REMLISP restituisce a DOS la memoria consumata da EXTLISP. Se EXTLISP è stato l'ultimo programma TSR caricato, questa procedura riporterà la memoria alla situazione prima di lanciare EXTLISP.

2.4 Funzioni automatiche - File "acad.lsp"

Ogni volta che inizia una sessione nell'Editore di Disegni di AutoCAD, AutoLISP carica il file "acad.lsp" (se presente). In questo file si possono riporre le definizioni di funzioni ricorrenti: esse verranno valutate automaticamente ogni volta che si inizia a editare un disegno. Per ulteriori informazioni a riguardo, rimandiamo alla funzione DEFUN nel Cap. 4 di questo manuale.

Capitolo 3

Esempio di programmazione

La potenza di AutoCAD risiede principalmente nella possibilità di personalizzarlo. Man mano che si acquista dimestichezza con AutoCAD, cresce anche l'esigenza di automatizzare determinate operazioni ricorrenti. Ciò è possibile grazie all'*architettura aperta* di AutoCAD, che permette di trasformarlo gradualmente in uno strumento di lavoro che risponde perfettamente alle esigenze specifiche di ogni utente. Si può, ad esempio, aggiungere ai menù di schermo, di tavoletta e dei pulsanti, sequenze di comandi frequentemente utilizzati; si possono definire nuovi tipi di linea, modelli di tratteggio, stili di testo, ecc. Le possibilità sono numerosissime.

AutoLISP, un'implementazione del linguaggio LISP compresa nel pacchetto opzionale ADE-3 di AutoCAD, rappresenta il mezzo migliore per potenziare AutoCAD. Infatti, con la possibilità di scrivere programmi in LISP, l'utente è in grado di aggiungere comandi personali ad AutoCAD e di modificare il programma, ha cioè a sua disposizione gli stessi mezzi usati dai nostri programmatori per lo sviluppo di AutoCAD.

In questo capitolo aggiungeremo un nuovo comando ad AutoCAD e questo esempio ci permetterà di spiegare il modo di procedere di AutoLISP. Anche se il comando che svilupperemo è concepito per essere impiegato nell'architettura dei giardini, i concetti trattati sono evidentemente validi per ogni tipo di applicazione.

3.1 Considerazioni preliminari

Presentando questo esempio di programmazione supponiamo che chi lo studia sia un utente pratico di AutoCAD, cioè pratico dei comandi e dei concetti fondamentali del programma. Inoltre supponiamo che abbia a disposizione un editore di testo con il quale può scrivere file ASCII.

Nell'esempio trattato in questo capitolo ricorreremo spesso a varie funzioni di AutoLISP; i Capitoli 4 e 5 sono interamente dedicati alla loro descrizione e spiegazione.

3.2 Obiettivo

L'obiettivo che ci siamo prefissati è quello di creare un nuovo comando AutoCAD che disegni un viale da giardino e lo riempia di piastrelle circolari in cemento. Il nuovo comando presenterà questa sequenza:

```
Comando:  VIALE
Punto di partenza del viale: (punto di partenza)
Punto finale del viale: (punto finale)
Mezza larghezza del viale: (numero)
Raggio delle piastrelle: (numero)
Spaziatura tra piastrelle: (numero)
```

in cui il *punto di partenza* e il *punto finale* specificano l'asse del viale. In seguito si ha la possibilità di specificare la mezza larghezza del viale, il raggio delle piastrelle e la spaziatura tra le piastrelle. La ragione per cui si usa la mezza larghezza piuttosto che la larghezza totale del

viale è dovuta al fatto che per specificarla si usa il cursore a linea elastica sullo schermo ancorato al punto iniziale del viale.

3.3 Convertire gradi in radianti

Nel nostro esempio faremo spesso uso di angoli e, dato che AutoLISP, come molti altri linguaggi di programmazione, specifica gli angoli in radianti mentre noi siamo abituati a specificarli in termini di gradi, definiremo una *funzione* che converte gradi in radianti. I radianti misurano gli angoli da zero a 2 * PI. Dapprima generiamo un file chiamato VIA.LSP con l'editore di testi, poi immettiamo il programma seguente:

```
;       Convertire angoli da gradi in radianti

(defun gir (a)
       (* pi (/ a 180.0))
)
```

Consideriamolo da vicino. Stiamo definendo una nuova funzione con la funzione DEFUN di AutoLISP. Questa nuova funzione si chiama GIR (sta per "gradi in radianti") e ha un argomento "A", cioè l'angolo in gradi, e dà come risultato l'espressione:

```
PI * (a / 180.0)
```

espressa in notazione LISP, che può essere letta nel modo seguente: "il prodotto di PI moltiplicato per il quoziente di A diviso per 180.0". PI è predefinito in AutoLISP e sta per 3.14159.... La riga introdotta da un punto e virgola è un *commento*. AutoLISP ignora qualsiasi testo su una riga che inizia con un punto e virgola.

Ora salviamo il file su disco e lanciamo AutoCAD per iniziare un nuovo disegno (il nome non importa, dato che non salveremo il disegno a sessione ultimata). Quando appare il messaggio "Comando:" di AutoCAD, carichiamo la funzione digitando:

```
Comando:  (load "via")
```

AutoLISP carica la funzione riferendosi ad essa con il nome che le avevamo assegnato ("GIR"). D'ora in poi, quando leggerete "Lanciamo AutoCAD e carichiamo il programma" in questo manuale, significa che ci riferiamo alla sequenza appena descritta.

Passiamo ora a verificare la funzione eseguendola con diversi valori. Dalla definizione di radianti data sopra, zero gradi dovrebbero corrispondere a zero radianti, quindi immettiamo:

```
Comando:  (gir 0)
```

in AutoCAD. Digitando una riga che inizia con una parentesi di apertura induciamo AutoCAD a passare la riga ad AutoLISP per la valutazione. In questo caso facciamo valutare la funzione GIR che abbiamo appena definito, dandole l'argomento zero. Dopo la valutazione della funzione, AutoCAD visualizza il risultato, quindi la riga che abbiamo digitato dovrebbe produrre la risposta:

```
0.0
```

Proviamo con 180 gradi. Se immettiamo:

```
Comando:  (gir 180)
```

dovremmo ottenere:

```
3.14159
```

Questo significa che 180 gradi è pari a PI radianti. Esaminando la funzione constatiamo che ciò corrisponde a quanto definito sopra.

Ora usciamo da AutoCAD digitando:

```
Comando:  USCIRE
Volete eliminare le modifiche apportate al disegno?  S
```

e rispondiamo:

```
0
```

al messaggio di richiesta del Menu Principale per tornare al messaggio di DOS. D'ora in poi, quando in questo testo figurerà "Usciamo da AutoCAD", si intenderà questo procedimento.

3.4 Ottenere dati relativi al viale

Il nostro comando VIALE deve chiedere all'utente di specificare la posizione in cui disegnare il viale, la sua larghezza, la dimensione delle piastrelle in cemento e la spaziatura fra di esse. Definiremo una funzione che sollecita l'immissione di tutti questi parametri e calcola i vari numeri che useremo in seguito.

Con l'editore di testo aggiungiamo a VIA.LSP le righe seguenti (per mettere in evidenza le righe aggiunte di volta in volta le abbiamo indicate con barre verticali):

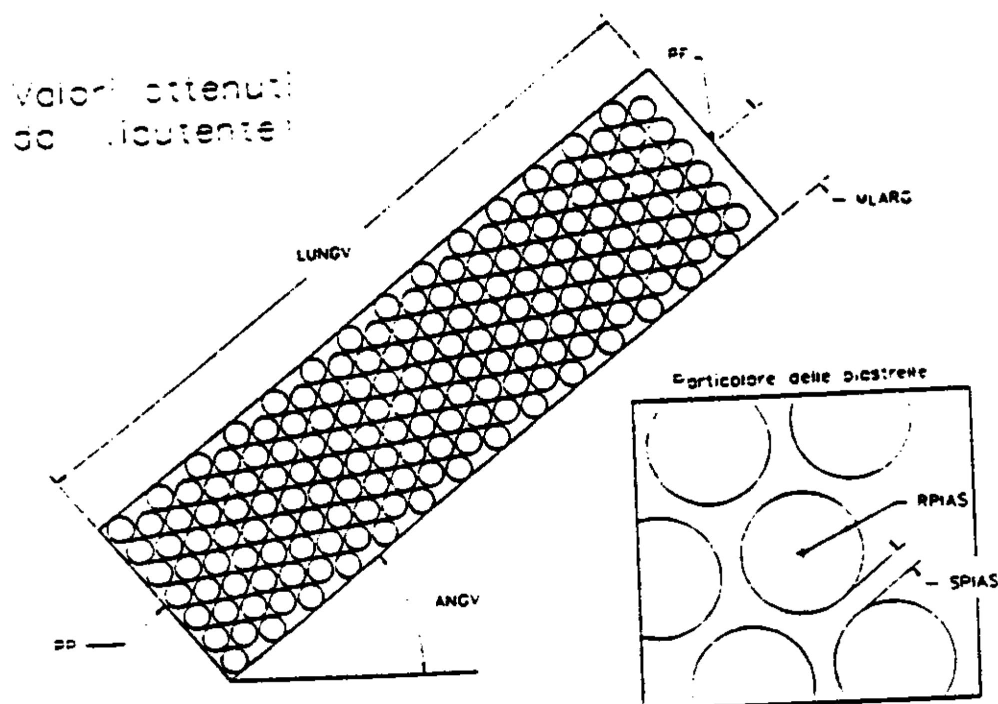
```
;       Convertire angoli da gradi in radianti
(defun gir (a)
        (* pi (/ a 180.0))
)
    |
    |   ;       Ottenere dati relativi al viale
    |
    |   (defun viautente ()
    |           (setq pp (getpoint "\nPunto di partenza del viale: "))
    |           (setq pf (getpoint "\nPunto finale del viale: "))
    |           (setq mlarg (getdist "\nMezza larghezza del viale: " pp))
    |           (setq rpias (getdist "\nRaggio piastrelle: " pp))
    |           (setq spias (getdist "\nSpaziatura tra piastrelle: " pp))
    |
    |           (setq angv (angle pp pf))
    |           (setq lungv (distance pp pf))
    |           (setq larg (* 2 mlarg))
    |           (setq angp90 (+ angv (gir 90)))        ; Angolo viale + 90 gradi
    |           (setq angm90 (- angv (gir 90)))        ; Angolo viale - 90 gradi
    |   )
```

Non è obbligatorio rientrare i paragrafi della funzione. Volendo si può addirittura digitare l'intera funzione senza andare mai a capo. Il rientrare i paragrafi e l'andare a capo hanno l'unico scopo di dare un'ordine interno alla funzione, in modo che sia più chiara e leggibile. Anche le parentesi iniziali e finali delle espressioni principali sono state allineate per evidenziarle e risulti perciò più difficile il dimenticare delle parentesi di chiusura.

Con ciò abbiamo definito una funzione chiamata VIAUTENTE. Questa funzione non possiede argomenti e sollecita l'utente ad immettere i valori richiesti per i vari parametri. La funzione SETQ dà a una variabile di AutoLISP un valore specifico. La prima funzione, SETQ, dà alla variabile PP (punto di partenza) il risultato della funzione GETPOINT. La funzione GETPOINT sollecita l'immissione di un punto da parte dell'utente. La stringa specifica il messaggio di richiesta che apparirà in AutoCAD. "\n" fa si che il messaggio venga visualizzato sulla riga successiva. Usiamo la funzione GETDIST per ottenere la mezza larghezza del viale, il raggio delle piastrelle e la loro spaziatura. Il secondo argomento della funzione GETDIST, PP, è il "punto base" della distanza, che è determinata da questo e dal punto finale del viale, immesso con l'aiuto del cursore a linea elastica ancorato al punto di partenza.

Dopo aver ottenuto i dati richiesti, AutoLISP calcola diverse variabili ricorrenti: ad ANGV viene assegnato l'angolo compreso fra il segmento che unisce il punto di partenza e quello finale e l'orizzontale; la funzione ANGLE restituisce questo angolo dopo l'immissione dei due punti; la variabile LUNGV ottiene la lunghezza del viale. Dati due punti, la funzione DISTANCE ne calcola la distanza. Visto che abbiamo specificato la mezza larghezza del viale, ne possiamo dedurre la larghezza totale. Infine calcoliamo e memorizziamo l'angolo del viale più o meno 90 gradi nelle variabili ANGP90 e ANGM90 rispettivamente (dato che in AutoLISP gli angoli sono in radianti, abbiamo usato la funzione GIR per convertire gradi in radianti prima di passare ai calcoli).

La figura illustra l'effetto delle variabili ottenute da VIAUTENTE.



A questo punto salviamo il programma aggiornato su disco, richiamiamo AutoCAD e carichiamo il programma. Passiamo ora a verificare la funzione. Dapprima attiviamo la funzione digitando:

```
Comando:   (viautente)
```

e alle richieste rispondiamo come segue:

```
Punto di partenza del viale:  2,2
Punto finale del viale:  9,8
Mezza larghezza del viale:  2
Raggio delle piastrelle:  .2
Spaziatura tra piastrelle:  .1
```

VIAUTENTE usa i valori forniti per calcolare le variabili addizionali e visualizza il risultato del suo ultimo calcolo (in questo caso, -0.86217, che corrisponde al valore di ANGM90 in radianti). Per ottenere il valore delle variabili specificate dalla funzione VIAUTENTE, basta immetterne il nome preceduto da un punto esclamativo (!). Ciò induce AutoCAD a valutare la variabile e a visualizzare il risultato ottenuto. Immettendo i comandi seguenti, dovremmo ottenere i risultati mostrati:

```
Comando:  !pp
(2.0 2.0 0.0)
Comando:  !pf
(9.0 8.0 0.0)
Comando:  !mlarg
2.0
Comando:  !larg
4.0
Comando:  !rpias
0.2
Comando:  !spias
0.1
Comando:  !angv
0.708626
Comando:  !lungv
9.21954
Comando:  !angp90
2.27942
Comando:  !angm90
-0.86217
```

Si noti che PP e PF sono restititi in forma di punti a tre dimensioni, in questo esercizio ignoreremo però la componente. ANGV, ANGP90 e ANGM90 sono rappresentati in radianti. Dopo aver verificato questi valori, usciamo da AutoCAD e torniamo a VIA.LSP nell'editore di testo.

3.5 Disegnare il contorno del viale

Dopo aver ottenuto i dati relativi al viale, passiamo ora a disegnarne il contorno. Aggiungiamo le linee indicate al nostro file VIA.LSP.

```
;        Convertire angoli da gradi in radianti
(defun gir (a)
        (*pi (/ a 180.0))
)

        ;        Ottenere dati relativi al viale

        (defun viautente ()
                (setq pp (getpoint "\nPunto di partenza del viale: "))
                (setq pf (getpoint "\nPunto finale del viale: "))
                (setq mlarg (getdist "\nMezza larghezza del viale: " pp))
                (setq rpias (getdist "\nRaggio piastrelle: " pp))
                (setq spias (getdist "\nSpaziatura tra piastrelle: " pp))

                (setq angv (angle pp pf))
                (setq lungv (distance pp pf))
                (setq larg (* 2 mlarg))
                (setq angp90 (+ angv (gir 90)))        ; Angolo viale + 90 gradi
                (setq angm90 (- angv (gir 90)))        ; Angolo viale - 90 gradi
        )

|       ;        Disegnare il contorno del viale
|
|       (defun discon ()
|               (command "plinea"
|                       (setq p (polar pp angm90 mlarg))
|                       (setq p (polar p angv lungv))
|                       (setq p (polar p angp90 larg))
|                       (polar p (+ angv (gir 180)) lungv)
|                       "chiuso"
|               )
|       )
```

Questa aggiunta definisce una funzione chiamata DISCON; essa usa il punto di partenza, l'angolo e la lunghezza del viale ottenuti dalla funzione VIAUTENTE, e disegna il contorno del viale con una polilinea. La funzione DISCON usa la funzione COMMAND per sottoporre comandi e dati ad AutoCAD. La funzione COMMAND è infatti il meccanismo con cui funzioni di AutoLISP sottomettono comandi ad AutoCAD per provocarne l'esecuzione. La funzione COMMAND può possedere un numero qualsiasi di argomenti e sottopone ciascuno di essi ad AutoCAD. Nel nostro caso, il comando dato ad AutoCAD è "plinea", cioè il comando che traccia polilinee.

In seguito specifichiamo i quattro vertici del rettangolo che forma il viale; ognuno di essi viene elaborato mediante la funzione POLAR ed è poi memorizzato nella variabile temporanea P. La funzione POLAR ha come primo argomento un punto, come secondo un angolo e come terzo argomento una distanza e restituisce un punto alla distanza e angolazione specificata dal punto di partenza. In questo caso calcoliamo i quattro punti legando geometricamente il viale al suo punto di partenza. Per concludere il comando, inviamo la stringa "chiuso" al comando PLINEA.

con la quale viene tracciato il quarto lato del viale e torna poi al messaggio "Comando:" di AutoCAD.

Per fare un controllo del funzionamento corretto di questa funzione, salviamo il file VIA.LSP aggiornato, richiamiamo AutoCAD con un nuovo disegno e carichiamo il file. Attiviamo dapprima la funzione di richiesta dati:

```
Comando:  (viautente)
```

e specifichiamo i valori come già fatto in precedenza. Poi richiamiamo la funzione DISCON:

```
Comando:  (discon)
```

La funzione darà ad AutoCAD i comandi definiti per disegnare il contorno e vedremo apparire il contorno del viale sul monitor. Dopodiché usciamo da AutoCAD.

3.6 Disegnare le piastrelle

Dopo aver sviluppato e verificato la funzione di richiesta dati e la funzione che disegna il contorno del viale, passiamo ora a riempire il viale con le piastrelle circolari. Ci troviamo nell'editore di testo, ciò che ci permette di aggiungere al programma le nuove righe indicate:

```
;       Convertire angoli da gradi in radianti
(defun gir (a)
        (*pi (/ a 180.0))
)
        ;       Ottenere dati relativi al viale

        (defun viautente ()
                (setq pp (getpoint "\nPunto di partenza del viale: "))
                (setq pf (getpoint "\nPunto finale del viale: "))
                (setq mlarg (getdist "\nMezza larghezza del viale: " pp))
                (setq rpias (getdist "\nRaggio piastrelle: " pp))
                (setq spias (getdist "\nSpaziatura tra piastrelle: " pp))

                (setq angv (angle pp pf))
                (setq lungv (distance pp pf))
                (setq larg (* 2 mlarg))
                (setq angp90 (+ angv (gir 90)))         ; Angolo viale + 90 gradi
                (setq angm90 (- angv (gir 90)))         ; Angolo viale - 90 gradi
        )

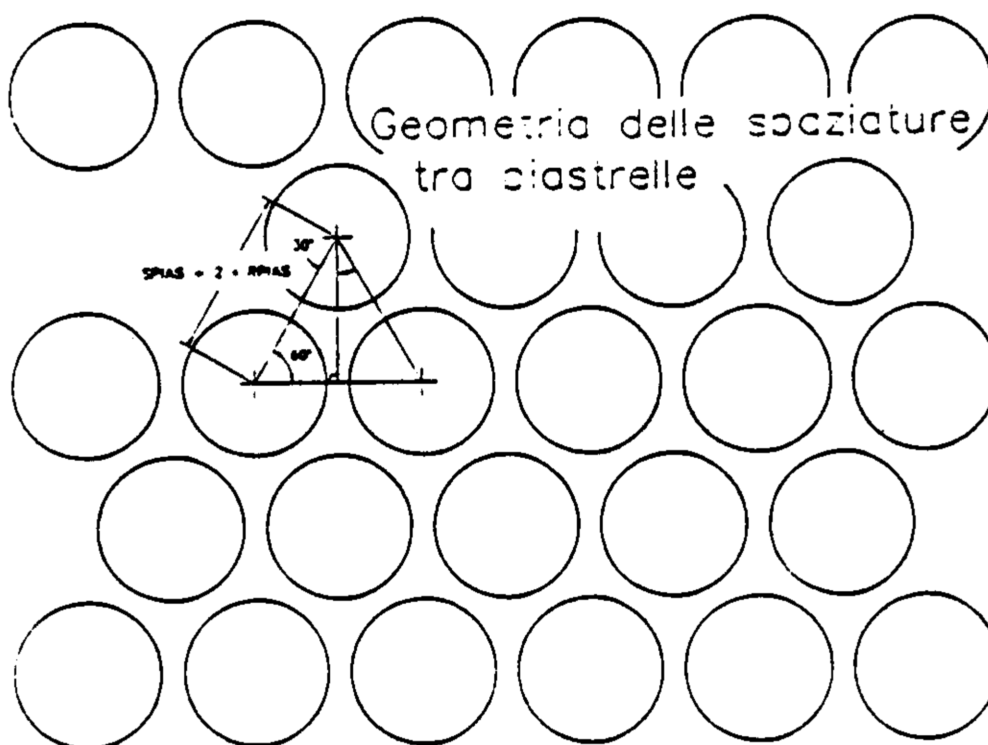
        ;       Disegnare il contorno del viale

        (defun discon ()
                (command "plinea"
                        (setq p (polar pp angm90 mlarg))
                        (setq p (polar p angv lungv))
                        (setq p (polar p angp90 larg))
                        (polar p (+ angv (gir 180)) lungv)
                        "chiuso"
                )
        )
```

```
;       Posizionare una fila di piastrelle data la lunghezza del viale
;       e se possibile sfalsarla
(defun disfila (lv sfalsa)
        (setq primap (polar pp angv lv))
        (setq pcpias (polar primap angp90 sfalsa))
        (setq plpias pcpias)
        (while (< (distance primap plpias) (- mlarg rpias))
                (command "cerchio" plpias rpias)
                (setq plpias (polar plpias angp90 (+ spias rpias rpias)))
        )
        (setq plpias (polar pcpias angm90 (+ spias rpias rpias)))
        (while (< (distance primap plpias) (- mlarg rpias))
                (command "cerchio" plpias rpias)
                (setq plpias (polar plpias angm90 (+ spias rpias rpias)))
        )
)

;       Disegnare le file di piastrelle
(defun dispias ()
        (setq plungv (+ rpias spias))
        (setq sfal 0.0)
        (while (<= plungv (- lungv rpias))
                (disfila plungv sfal)
                (setq plungv (+ plungv (* (+ spias rpias rpias) (sin (gir 60)))))
                (if (= sfal 0.0)
                        (setq sfal (* (+ spias rpias rpias) (cos (gir 60))))
                        (setq sfal 0.0)
                )
        )
)
```

Per comprendere meglio l'effetto di queste funzioni facciamo riferimento alla figura seguente. La funzione DISFILA disegna una fila di piastrelle con una data spaziatura lungo il viale specificato dal suo primo argomento. sfalsando la fila successiva di una distanza data dal secondo argomento. Per coprire razionalmente la superficie a disposizione, sfalsiamo ora le piastrelle una rispetto all'altra.



DISFILA trova la posizione della prima piastrella nella fila usando POLAR per spostarsi lungo il viale della distanza data dal primo argomento e usando nuovamente POLAR per spostarsi perpendicolarmente al viale per la sfalsatura. In seguito usa la funzione WHILE per continuare a disegnare cerchi finché incontra il bordo del viale. SETQ alla fine del ciclo di WHILE si sposta nella posizione della piastrella successiva con una spaziatura di due volte il raggio della piastrella più la spaziatura intermedia.

Il secondo ciclo di WHILE disegna le file di piastrelle nell'altra direzione finché viene incontrato il bordo opposto del viale.

DISPIAS è la funzione che richiama ripetutamente DISFILA affinché disegni tutte le file di piastrelle. Il suo ciclo WHILE avanza lungo il viale richiamando DISFILA per ogni fila. Le piastrelle di file adiacenti formano triangoli equilateri come mostrato nella figura. I lati di questi triangoli sono uguali al doppio del raggio delle piastrelle più la spaziatura intermedia. Quindi questo valore moltiplicato per il seno di 60 gradi dà la distanza tra le file e lo stesso valore moltiplicato per il coseno di 60 gradi dà il valore di cui vengono sfalsate le file pari rispetto a quelle dispari.

Si noti il modo in cui viene usata la funzione IF in DISFILA per sfalsare le file pari: la funzione IF verifica il suo primo argomento ed esegue il secondo argomento se è vero, se invece è falso, esegue il terzo. Nel nostro caso, quando OFF è uguale a zero, la fila viene sfalsata del valore che si ottiene moltiplicando la distanza tra i centri delle piastrelle per il coseno di 60 gradi. Quando OFF è diverso da zero, il valore della sfalsatura è zero.

Per fare una verifica di questa funzione, salviamo il file, richiamiamo AutoCAD e carichiamo il programma. Poi digitiamo:

```
Comando:  (viautente)
```

e rispondiamo alle richieste come già visto. Immettiamo:

```
Comando:  (discon)
```

e viene tracciato il contorno, come già visto. Digitiamo infine:

```
Comando:  (dispias)
```

e tutte le piastrelle dovrebbero essere disegnate entro i bordi del viale.

3.7 Aggiungere il comando ad AutoCAD

Siamo ora in grado di combinare le varie sequenze sviluppate finora per farne un comando AutoCAD. Se in AutoLISP definiamo una funzione con il nome C:XXXX, digitando XXXX (a condizione che XXXX non sia un comando già definito in AutoCAD) richiamiamo la funzione. Per completare l'implementazione del comando VIALE, definiamo la funzione C:VIALE che ci permette di digitare semplicemente VIALE in un momento qualsiasi dopo aver caricato VIA.LSP per eseguire il comando che traccia un viale da giardino.

Aggiungiamo ora le righe indicate al file VIA.LSP, richiamiamo AutoCAD e carichiamo il programma.

```
;        Convertire angoli da gradi in radianti
(defun gir (a)
         (*pi (/ a 180.0))
)
          ;        Ottenere dati relativi al viale

          (defun viautente ()
                  (setq pp (getpoint "\nPunto di partenza del viale: "))
                  (setq pf (getpoint "\nPunto finale del viale: "))
                  (setq mlarg (getdist "\nMezza larghezza del viale: " pp))
                  (setq rpias (getdist "\nRaggio piastrelle: " pp))
                  (setq spias (getdist "\nSpaziatura tra piastrelle: " pp))

                  (setq angv (angle pp pf))
                  (setq lungv (distance pp pf))
                  (setq larg (* 2 mlarg))
                  (setq angp90 (+ angv (gir 90)))        ; Angolo viale + 90 gradi
                  (setq angm90 (- angv (gir 90)))        ; Angolo viale - 90 gradi
          )

          ;        Disegnare il contorno del viale

          (defun discon ()
                  (command "plinea"
                           (setq p (polar pp angm90 mlarg))
                           (setq p (polar p angv lungv))
                           (setq p (polar p angp90 larg))
                           (polar p (+ angv (gir 180)) lungv)
                           "chiuso"
                  )
```

```
        )

        ;       Posizionare una fila di piastrelle data la lungnezza del viale
        ;       e se possibile sfalsarla

        (defun disfila (lv sfalsa)
                (setq primap (polar pp angv lv))
                (setq pcpias (polar primap angp90 sfalsa))
                (setq pipias pcpias)
                (while (< (distance primap pipias) (- mlarg rpias))
                        (command "cerchio" pipias rpias)
                        (setq pipias (polar pipias angp90 (+ spias rpias rpias)))
                )
                (setq pipias (polar pcpias angm90 (+ spias rpias rpias)))
                (while (< (distance primap pipias) (- mlarg rpias))
                        (command "cerchio" pipias rpias)
                        (setq pipias (polar pipias angm90 (+ spias rpias rpias)))
                )
        )

        ;       Disegnare le file di piastrelle

        (defun dispias ()
                (setq plungv (+ rpias spias))
                (setq sfal 0.0)
                (while (<= plungv (- lungv rpias))
                        (disfila plungv sfal)
                        (setq plungv (+ plungv (* (+ spias rpias rpias) (sin (gir 60)))))
                        (if (= sfal 0.0)
                                (setq sfal (* (+ spias rpias rpias) (cos (gir 60))))
                                (setq sfal 0.0)
                        )
                )
        )

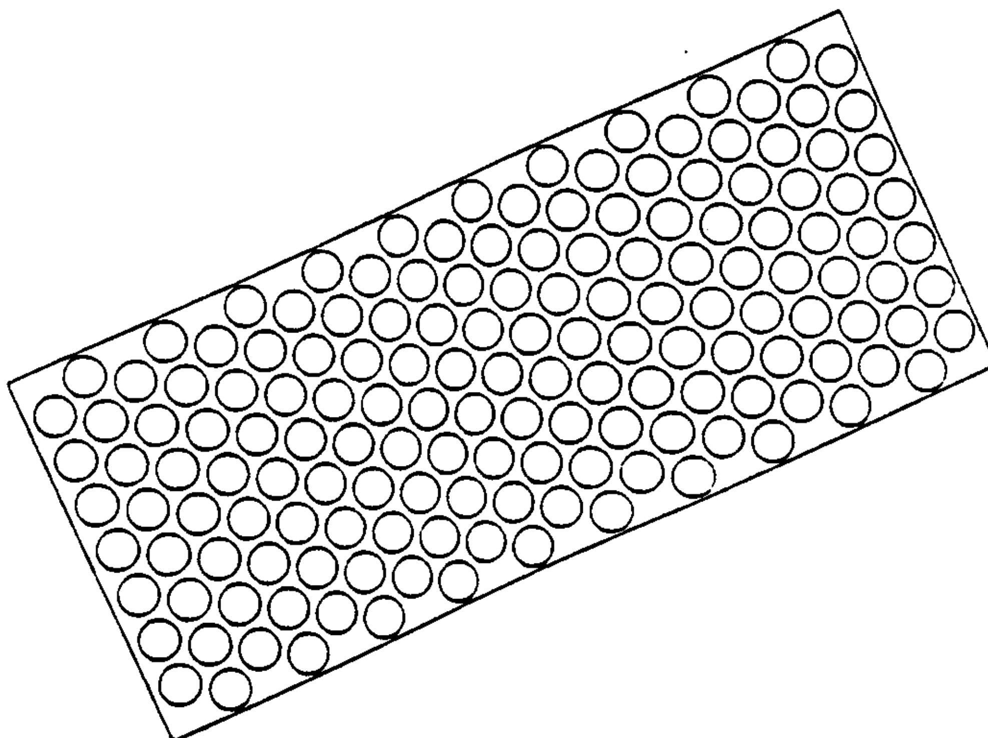
        ;       Eseguire il comando, richiamando le sue funzioni

        (defun C:VIALE ()
                (visutente)
                (discon)
                (dispias)
        )
```

Aggiungendo una funzione chiamata C:VIALE abbiamo aggiunto il comando VIALE ad AutoCAD. Possiamo fare una verifica del comando immettendo:

```
Comando:  VIALE
Punto di partenza del viale:  2,2
Punto finale del viale:  9,8
Mezza larghezza del viale:  2
Raggio delle piastrelle:  .2
Spaziatura tra piastrelle:  .1
```

Questa sequenza disegna la figura che segue.



3.8 Eliminare la ripetizione dei comandi

Durante l'esecuzione del comando VIALE, tutti i comandi che questo sottopone ad AutoCAD vengono ripetuti nell'area dei messaggi e tutti i punti che seleziona sono marcati sullo schermo mediante piccole croci. Dopo che il programma è stato messo a punto, la visualizzazione di questi dati può essere disattivata affinché il comando implementato si presenti come un comando AutoCAD. Aggiungiamo le righe al file VIA.LSP per sopprimere la ripetizione dei comandi e le croci di marcatura:

```
;       Convertire angoli da gradi in radianti

(defun gir (a)
        (* pi (/ a 180.0))
)

;       Ottenere dati relativi al viale

(defun viautente ()
        (setq pp (getpoint "\nPunto di partenza del viale: "))
        (setq pf (getpoint "\nPunto finale del viale: "))
        (setq mlarg (getdist "\nMezza larghezza del viale: " pp))
        (setq rpias (getdist "\nRaggio piastrelle: " pp))
        (setq spias (getdist "\nSpaziatura tra piastrelle: " pp))

        (setq angv (angle pp pf))
```

```
        (setq lungv (distance pp pf))
        (setq larg (* 2 mlarg))
        (setq angp90 (+ angv (gir 90)))          ; Angolo viale + 90 gradi
        (setq angm90 (- angv (gir 90)))          ; Angolo viale - 90 gradi
)

;       Disegnare il contorno del viale

(defun discon ()
        (command "plinea"
                (setq p (polar pp angm90 mlarg))
                (setq p (polar p angv lungv))
                (setq p (polar p angp90 larg))
                (polar p (+ angv (gir 180)) lungv)
                "chiuso"
        )
)

;       Posizionare una fila di piastrelle data la lungnezza del viale
;       e se possibile sfalsarla
(defun disfila (lv sfalsa)
        (setq primap (polar pp angv lv))
        (setq pcpias (polar primap angp90 sfalsa))
        (setq pipias pcpias)
        (while (< (distance primap pipias) (- mlarg rpias))
                (command "cerchio" pipias rpias)
                (setq pipias (polar pipias angp90 (+ spias rpias rpias)))
        )
        (setq pipias (polar pcpias angm90 (+ spias rpias rpias)))
        (while (< (distance primap pipias) (- mlarg rpias))
                (command "cerchio" pipias rpias)
                (setq pipias (polar pipias angm90 (+ spias rpias rpias)))
        )
)

;       Disegnare le file di piastrelle
(defun dispias ()
        (setq plungv (+ rpias spias))
        (setq sfal 0.0)
        (while (<= plungv (- lungv rpias))
                (disfila plungv sfal)
                (setq plungv (+ plungv (* (+ spias rpias rpias) (sin (gir 60)))))
                (if (= sfal 0.0)
                        (setq sfal (* (+ spias rpias rpias) (cos (gir 60))))
                        (setq sfal 0.0)
                )
        )
)

;       Eseguire il comando, richiamando le sue funzioni

(defun C:VIALE ()
        (viautente)
        (setq ptinis (getvar "blipmode"))
```

```
|          (setq ecocoms (getvar "cmdecho"))
|          (setvar "blipmode" 0)
|          (setvar "cmdecho" 0)
           (discon)
           (dispias)
|          (setvar "blipmode" ptinis)
|          (setvar "cmdecho" ecocoms)
     )
```

Usiamo la funzione GETVAR per ottenere i valori correnti delle variabili BLIPMODE e CMECHO di AutoCAD. Queste variabili sono salvate mediante SETQ e ECOCOMS. Usiamo poi la funzione SETVAR per azzerare le due variabili, disattivando la visualizzazione dei punti di riferimento e delle sequenze di comando. E' da notare che azzeriamo queste variabili solo dopo aver ottenuto i dati dall'utente tramite VIAUTENTE. I punti di riferimento possono infatti rivelarsi utili per avere una conferma dell'immissione dei dati.

Quando il disegno del viale è terminato, usiamo la funzione SETVAR per ridare alle due variabili i loro valori originali.

Infine salviamo il file, richiamiamo AutoCAD e proviamo ad immettere il comando VIALE, specificando parametri diversi sia dalla tastiera che con il puntatore.

3.9 Riassunto

Con un investimento di tempo relativamente piccolo, abbiamo aggiunto un nuovo comando ad AutoCAD. Con altri sistemi CAD avremmo dovuto accedere al codice di sorgente del sistema, essere programmatori esperti e conoscere a fondo un'enorme quantità di nozioni per ottenere lo stesso risultato. L'architettura aperta di AutoCAD e di AutoLISP permettono di usufruire delle prestazioni che la maggior parte dei fornitori di CAD riservano ai loro programmatori.

Questo esempio può servire da primo esercizio per iniziare a lavorare con AutoLISP. In seguito si può provare a modificare o ad ampliare il comando VIALE, facendogli disegnare, ad esempio, piastrelle quadrate o esagonali. Un'applicazione più avanzata consisterebbe nel creare un nuovo comando che sollecita l'immissione di un punto centrale e di un'area per disegnare un quadrato che abbia l'area specificata e che la riempia di piastrelle.

I capitoli che seguono presentano dettagliatamente la sintassi e il modo di operare di ogni funzione di AutoLISP. In questo capitolo ci siamo limitati a presentare una breve descrizione dell'effetto di alcune funzioni, ma provando a creare e a modificare altre funzioni scoprirete voi stessi le innumerevoli possibilità che AutoLISP offre.

Capitolo 4

Funzioni di AutoLISP

AutoLISP contiene numerose funzioni predefinite. Ogni funzione è richiamata con il suo nome (in lettere maiuscole o minuscole) quale primo elemento di un lista, con gli argomenti relativi ad essa (se presenti) come elementi successivi della lista.

Questo capitolo presenta tutte le funzioni essenziali di AutoLISP in ordine alfabetico. L'indice analitico di questo manuale raggruppa le funzioni per facilitarne la consultazione. Molte funzioni sono standard, cioè si ritrovano in ogni implementazione del linguaggio di programmazione LISP, mentre altre funzioni sono tipiche dell'ambiente grafico interattivo di AutoCAD. Alcune funzioni avanzate verranno descritte nei capitoli seguenti.

4.1 La variabile di sistema FLATLAND - Compatibilità con versioni precedenti

AutoLISP Release 10 gestisce tutte le nuove funzioni 3D di AutoCAD pur restando compatibile con tutte le routine scritte per versioni precedenti di AutoCAD. La variabile di sistema FLATLAND è stata aggiunta proprio per garantire questa compatibilità. Quando FLATLAND ha valore 0, le modifiche 3D sono attive, se si assegna invece a FLATLAND un valore diverso da 0, la funzioni si comportano come nella versione 9 di AutoLISP. L'impostazione di FLATLAND influisce sulle seguenti funzioni di AutoLISP:

DISTANCE	GRREDAD	POLAR
ENTGET	INITGET	TBLNEXT
GETDIST	INTERS	TBLSEARCH
GETPOINT	OSNAP	

La maniera in cui FLATLAND influisce su queste funzioni è riportata sotto la descrizione di ogni funzione.
FLATLAND è un ausilio temporaneo alla conversione e verrà eliminata nella prossima versione di AutoCAD.

4.2 (+ <numero> <numero>...)

Questa funzione restituisce la somma di tutti i numeri, che possono essere reali o interi. Se tutti i numeri sono interi, il risultato sarà un intero; se alcuni dei numeri sono reali, i numeri interi verranno convertiti in reali e il risultato sarà un numero reale. Esempio:

```
(+ 1 2)          restituisce   3
(+ 1 2 3 4.5)    restituisce   10.5
(+ 1 2 3 4.0)    restituisce   10.0
```

4.3 (- <numero> <numero>...)

Questa funzione sottrae il secondo <numero> dal primo e restituisce la differenza. Se sono dati più di due numeri, la somma del secondo più gli altri fino all'ultimo viene sottratta dal primo e viene restituito il valore finale. Se è dato un solo <numero>, il risultato è dato dalla sottrazione

di questo da zero. Questa funzione può essere usata con numeri interi o reali e valgono le regole standard di conversione. Esempio:

`(- 50 40)`	restituisce	10
`(- 50 40.0 2)`	restituisce	8.0
`(- 50 40.0 2.5)`	restituisce	7.5
`(- 8)`	restituisce	-8

4.4 (* <numero> <numero>...)

Questa funzione restituisce il prodotto di tutti i numeri. Questa funzione può essere usata con numeri interi o reali e valgono le regole standard di conversione. Esempio:

`(* 2 3)`	restituisce	6
`(* 2 3 4.0)`	restituisce	24.0
`(* 3 -4.5)`	restituisce	-13.5

4.5 (/ <numero> <numero>...)

Questa funzione divide il primo <numero> per il secondo e restituisce il quoziente. Se sono dati più di due numeri, il primo <numero> viene diviso per il prodotto del secondo con gli altri fino all'ultimo e viene restituito il quoziente finale. Questa funzione può essere usata con numeri interi o reali e valgono le regole standard di conversione. Esempi:

`(/ 100 2)`	restituisce	50
`(/ 100 2.0)`	restituisce	50.0
`(/ 100 20.0 2)`	restituisce	2.5
`(/ 100 20 2)`	restituisce	2
`(/ 135 360)`	restituisce	0
`(/ 135 360.0)`	restituisce	0.375

4.6 (= <atomo> <atomo>...)

Questa è la funzione "uguale a", che restituisce T se tutti gli atomi specificati sono numericamente uguali o NIL se non lo sono. Questa funzione è valida anche per numeri e stringhe. Esempi:

`(= 4 4.0)`	restituisce	T
`(= 20 388)`	restituisce	NIL
`(= 2.4 2.4 2.4)`	restituisce	T
`(= 499 499 500)`	restituisce	NIL
`(= "mio" "mio")`	restituisce	T
`(= "mio" "tuo")`	restituisce	NIL

4.7 (/= <atom1> <atom2>)

Questa è la funzione "diverso da", che restituisce T se il primo atomo non è numericamente uguale al secondo o NIL in caso contrario. Se sono presenti più di due atomi, la funzione rimane indefinita. Per esempio:

`(/= 10 20)`	restituisce	T
`(/= "tu" "tu")`	restituisce	NIL
`(/= 5.43 5.44)`	restituisce	T

4.8 (< <atomo> <atomo>...)

Questa è la funzione "minore di", che restituisce T se il primo <atomo> è minore del secondo o NIL in caso contrario. Se sono dati più di due atomi, viene restituito T se ogni atomo è minore dell' <atomo> alla sua destra. Ad esempio:

`(< 10 20)`	restituisce	T
`(< "b" "c")`	restituisce	T
`(< 357 33.2)`	restituisce	NIL
`(< 2 3 88)`	restituisce	T
`(< 2 3 4 4)`	restituisce	NIL

4.9 (<= <atomo> <atomo>...)

Questa è la funzione relazionale "minore di o uguale a", che restituisce T se il primo <atomo> è minore o uguale al secondo o NIL in caso contrario. Se sono presenti più di due atomi, viene restituito T se ogni atomo è minore o uguale all'<atomo> alla sua destra.

`(<= 10 20)`	restituisce	T
`(<= "b" "b")`	restituisce	T
`(<= 357 33.2)`	restituisce	NIL
`(<= 2 9 9)`	restituisce	T
`(<= 2 9 4 5)`	restituisce	NIL

4.10 (> <atomo> <atomo>...)

Questa è la funzione "maggiore di", che restituisce T se il primo <atomo> è maggiore del secondo o NIL in caso contrario. Se sono presenti più di due atomi, viene restituito T se ogni atomo è maggiore dell'<atomo> alla sua destra. Esempi:

`(> 120 17)`	restituisce	T
`(> "c" "b")`	restituisce	NIL
`(> 3.5 1792)`	restituisce	NIL
`(> 77 4 2)`	restituisce	T
`(> 77 4 4)`	restituisce	NIL

4.11 (>= <atomo> <atomo>...)

Questa è la funzione "maggiore di o uguale a", che restituisce T se il primo <atomo> è maggiore di o uguale al secondo o NIL in caso contrario. Se sono presenti più di due atomi, viene restituito T se ogni atomo è maggiore di o uguale all'<atomo> alla sua destra. Esempi:

```
(>= 120 17)         restituisce   T
(>= "c" "c")        restituisce   T
(>= 3.5 1792)       restituisce   NIL
(>= 77 4 4)         restituisce   T
(>= 77 4 9)         restituisce   NIL
```

4.12 (~ <numero>)

Questa funzione restituisce il NOT (complemento di uno) del <numero> a livello di bit . <numero> deve essere un intero. Ad esempio:

```
(~ 3)               restituisce   -4
(~ 100)             restituisce   -101
(~ -4)              restituisce   3
```

4.13 (1+ <numero>)

Questa funzione restituisce <numero> aumentato di 1 (incrementato). <numero> può essere reale o intero. Esempi:

```
(1+ 5)              restituisce   6
(1+ -17.5)          restituisce   -16.5
```

4.14 (1- <numero>)

Questa funzione restituisce <numero> diminuito di 1 (decrementato). <numero> può essere reale o intero. Esempi:

```
(1- 5)              restituisce   4
(1- -17.5)          restituisce   -18.5
```

4.15 (abs <numero>)

Questa funzione restituisce il valore assoluto del <numero>. <numero> può essere reale o intero. Ad esempio:

```
(abs 100)           restituisce   100
(abs -100)          restituisce   100
(abs -99.25)        restituisce   99.25
```

4.16 (and <espressione>...)

Questa funzione restituisce l'AND logico di una lista di espressioni. Essa interrompe ogni ulteriore valutazione e restituisce NIL se una delle espressioni produce NIL, in caso contrario restituisce T. Determinando per esempio quanto segue:

```
(setq a 103)
(setq b NIL)
(setq c "stringa")
```

allora:

```
(and 1.4 a c)        restituisce   T
(and 1.4 a b c)      restituisce   NIL
```

4.17 (angle <pt1> <pt2>)

Questa funzione restituisce l'angolo di una linea retta passante dal punto <pt1> al punto <pt2> entrambi dell'UCS. L'angolo è misurato in radianti a partire dall'asse X del piano di costruzione corrente e crescente in direzione antioraria. Se vengono forniti punti 3D, verranno proiettati sul piano di costruzione corrente. Esempio:

```
(angle '(1.0 1.0) '(1.0 4.0))      restituisce   1.5708
(angle '(5.0 1.33) '(2.4 1.33))    restituisce   3.14159
```

4.18 (angtos <angolo> [<modo> [<precisione>]])

La funzione valuta <angolo> (un numero reale, in radianti) e lo restituisce trasformato in una stringa in accordo con le impostazioni di <modo>, <precisione> e della variabile di quotatura DIMZP. Gli argomenti <modo> e <precisione> sono interi che determinano il tipo di unità angolare e il suo grado di precisione. I valori validi per <modo> sono elencati qui sotto:

Modo ANGTOS	Formato di editazione
0	Gradi
1	Gradi/minuti/secondi
2	Gradi centesimali
3	Radianti
4	unità topografiche

L'argomento <precisione> è un intero che seleziona il numero di decimali richiesti dalla precisione desiderata. <modo> e <precisione> corrispondono alle variabili di sistema di AutoCAD, AUNITS e AUPREC. Se si omettono gli argomenti, verranno usate le impostazioni correnti di AUNITS e AUPREC.

Ad esempio, data DIMZP =0 e le impostazioni seguenti:

```
(setq pt1 '(5.0 1.33))
(setq pt2 '(2.4 1.33))
(setq a (angle pt1 pt2))
```

allora:

(angtos a 0 0)	restituisce	"180"
(angtos a 0 4)	restituisce	"180.0000"
(angtos a 1 4)	restituisce	"180d0'0""
(angtos a 3 4)	restituisce	"3.1416r"
(angtos a 4 2)	restituisce	"0"

ANGTOS accetta un argomento negativo per <angolo> ma lo trasforma sempre in un valore positivo tra 0 e 2 pi greco prima di eseguire la conversione richiesta. Ad esempio:

(angtos 0.785398 0 4)	restituisce	"45.0000"
(angtos -0.785398 0 4)	restituisce	"315.0000"

4.19 (append <espressione>...)

Questa funzione valuta un numero qualsiasi di liste (<espressione>) e dà come risultato un'unica lista. Esempi:

(append '(a b) '(c d))	restituisce	(A B C D)
(append '((a) (b)) '((c) (d)))	restituisce	((A) (B) (C) (D))

APPEND richiede che i suoi argomenti siano liste.

4.20 (apply <funzione> <lista>)

APPLY esegue la <funzione> con gli argomenti dati da <lista>. Esempi:

(apply '+ '(1 2 3))	restituisce	6
(apply 'strcat '("a" "b" "c"))	restituisce	"abc"

APPLY funziona sia con funzioni predefinite (subrs) che con funzioni definite dall'utente (create con DEFUN o LAMBDA).

4.21 (ascii <stringa>)

Questa funzione restituisce la conversione del primo carattere della <stringa> nel rispettivo codice ASCII (un intero) ed è simile alla funzione ASC in BASIC. Esempi:

(ascii "A")	restituisce	65
(ascii "a"	restituisce	97
(ascii "BIG")	restituisce	66

4.22 (assoc <elemento> <alista>)

Questa funzione usa il suo primo argomento <elemento> come chiave, cerca la chiave nella lista associativa <alista> e restituisce l'ingresso di <alista>. Se <elemento> non è reperibile come chiave in <alista>, ASSOC restituisce NIL. Supponendo che la lista "al", ad esempio, sia definita come segue:

```
((scatola) (larg 3) (altez 4.7263) (profond 5))
```

allora:

```
(assoc 'altez al)          restituisce   (ALTEZ 4.7263)
(assoc 'peso al)           restituisce   NIL
```

Le liste associative sono usate frequentemente per salvare dati ai quali si può accedere mediante una chiave. Ciò è simile alle serie o strutture di altri linguaggi di programmazione. La funzione SUBST descritta più avanti in questo capitolo, permette di rimpiazzare il valore associato ad una chiave in una lista associativa.

4.23 (atan <num1> [<num2>])

Se <num2> non è presente, ATAN restituisce l'arcotangente di <num1> in radianti. <num1> può essere negativo; gli angoli restituiti possono variare da -*pi*/2.0 a + *pi*/2.0 radianti. Esempi:

```
(atan 0.5)                 restituisce   0.463648
(atan 1.0)                 restituisce   0.785398
(atan -1.0)                restituisce   -0.785398
(angtos (atan -1.0) 0 4)   restituisce   "315.0000"
```

Se sono presenti sia <num1> che <num2>, viene restituito l'arcotangente di <num1>/<num2> in radianti. Se <num2> è zero, viene restituito un angolo di + o - 1.570796 radianti (90 gradi o -90 gradi) a seconda del segno di <num1>. Esempi:

```
(atan 2.0 3.0)                   restituisce   0.588003
(angtos (atan 2.0 3.0) 0 4)      restituisce   "33.6901"
(atan 2.0 -3.0)                  restituisce   2.55359
(angtos (atan 2.0 -3.0) 0 4)     restituisce   "146.3099"
(atan -2.0 3.0)                  restituisce   -0.588003
(atan -2.0 -3.0)                 restituisce   -2.55359
(atan 1.0 0.0)                   restituisce   1.5708
(angtos (atan 1.0 0.0) 0 4)      restituisce   "90.0000"
(atan -0.5 0.0)                  restituisce   -1.5708
(angtos (atan -0.5 0.0) 0 2)     restituisce   "270.00"
```

4.24 (atof <stringa>)

Questa funzione restituisce la conversione di una stringa in un numero reale. Ad esempio:

```
(atof "97.1")              restituisce   97.1
(atof "3")                 restituisce   3.0
```

4.25 (atoi <stringa>)

Questa funzione restituisce la conversione di una stringa in un numero intero.

```
(atoi "97")        restituisce   97
(atoi "3")         restituisce   3
(atoi "3.9")       restituisce   3
```

4.26 (atom <elemento>)

Questa funzione restituisce NIL se <elemento> è una lista, in caso contrario restituisce T. Se un elemento non è una lista è considerato un atomo. Determinando per esempio quanto segue:

```
(setq a '(x y z))
(setq b 'a)
```

allora:

```
(atom 'a)          restituisce   T
(atom a)           restituisce   NIL
(atom 'b)          restituisce   T
(atom b)           restituisce   T
(atom '(a b c))    restituisce   NIL
```

Alcuni dialetti di LISP differiscono nell'interpretazione di ATOM, quindi è necessario verificare i codici convertiti.

4.27 (Boole <funz> <int1> <int2>...)

Questa è una funzione Booleana generale a livello di bit. <funz> è un intero tra 0 e 15 che rappresenta una delle 16 funzioni Booleane possibili in due variabili. Gli argomenti interi successivi sono combinati logicamente e sono basati su questa funzione e la tabella a destra.

int1	int2	funz bit
0	0	8
0	1	4
1	0	2
1	1	1

Ogni bit di <int1> è accoppiato con il bit corrispondente di <int2> selezionando una riga della tabella. Il bit risultante è 0 o 1, a seconda dell'impostazione del bit di <funz> corrispondente alla sua riga nella tabella. Se il bit appropriato è definito in <funz>, il bit risultante è 1, altrimenti 0.

Alcuni dei valori di <funz> sono equivalenti alle operazioni Booleane standard, AND, OR, XOR e NOT, elencate qui di seguito:

Funz.	Operazione	Bit risultante è 1, se...
1	AND	entrambi i bit d'ingresso sono 1
6	XOR	solo uno dei due bit d'ingresso e 1
7	OR	entrambi o uno dei due bit d'ingresso e 1
8	NOT	entrambi i bit d'ingresso sono 0 (complemento di 1)

Ad esempio:

```
(Boole 1 12 5)
```

specifica un AND logico dei valori 12 e 5. Il risultato è 4. In modo analogo:

```
(Boole 6 6 5)
```

specifica un XOR logico dei valori 6 e 5, restituendo 3.

Si possono usare altri valori di <funz> per eseguire altre operazioni Booleane che non hanno nomi standard. Per esempio, se <funz> è 4, i bit risultanti sono determinati se i bit corrispondenti sono definiti in <int2> e <int1>. Quindi:

```
(Boole 4 3 14)
```

restituirebbe 12.

4.28 (boundp <atomo>)

Questa funzione restituisce T se <atomo> ha un valore legato ad esso (indipendentemente dallo scopo). Se nessun valore è legato a <atomo> (o se è legato a NIL), viene restituito NIL. Determinando per esempio quanto segue:

```
(setq a 1)
(setq b NIL)
```

allora:

```
(boundp 'a)          restituisce     T
(boundp 'b)          restituisce     NIL
```

4.29 caar, cadr, cddr, cadar, ecc.

AutoLISP gestisce concatenazioni di CAR e CDR fino a quattro livelli di profondità. Ad esempio:

```
(setq x' ((a b) cd)
```

allora:

`(caar x)`	equivale a	`(car (car x))`	che restituisce A
`(cdar x)`	equivale a	`(cdr (car x))`	che restituisce (B)
`(cadar x)`	equivale a	`(car (cdr (car x)))`	che restituisce B
`(cadr x)`	equivale a	`(car (cdr x))`	che restituisce C
`(cddr x)`	equivale a	`(cdr (cdr x))`	che restituisce (D)
`(caddr x)`	equivale a	`(car (cdr (cdr x)))`	che restituisce D

In AutoLISP, CADR è usato frequentemente per ottenere la coordinata Y di un punto bidimensionale o tridimensionale (cioè il secondo elemento di una lista di due o tre numeri reali). Analogamente CADDR può essere utilizzato per ottenere la componente Z di un punto a tre dimensioni. Ad esempio, posti i seguenti valori:

```
(setq pt2 '(5.2 1.0))        (un punto bidimensionale)
(setq pt3 '(5.2 1.0 3.0)     (un punto tridimensionale)
```

allora:

`(car pt2)`	restituisce	5.25
`(cadr pt2)`	restituisce	1.0
`(caddr pt2)`	restituisce	NIL
`(car pt3)`	restituisce	5.25
`(cadr pt3)`	restituisce	1.0
`(caddr pt3)`	restituisce	3.0

4.30 (car <lista>)

Questa funzione restituisce il primo elemento di <lista>. Se la <lista> è vuota, viene restituito NIL. Esempi:

`(car '(a b c))`	restituisce	A
`(car '((a b) c))`	restituisce	(A B)
`(car ' ())`	restituisce	NIL

4.31 (cdr <lista>)

Questa funzione restituisce una lista contenente tutti gli elementi della <lista> eccetto il primo. Se la <lista> è vuota, viene restituito NIL. Esempi:

`(cdr '(a b c))`	restituisce	(B C)
`(cdr '((a b) c))`	restituisce	(C)
`(cdr '())`	restituisce	NIL

Se l'argomento <lista> è costituito da una coppia puntata (vedi quanto riportato sotto CONS), CDR restituisce il secondo elemento senza comprenderlo all'interno di una lista. Ad esempio:

`(cdr '(a . b))`	restituisce	B
`(cdr '(1 . "Testo"))`	restituisce	"Testo"

4.32 (chr <numero>)

Questa funzione restituisce la conversione di un intero rappresentante un caratttere ASCII in una stringa di un carattere (simile alla funzione CHR$ in BASIC). Esempi:

```
(chr 65)        restituisce     "A"
(chr 66)        restituisce     "B"
(chr 97)        restituisce     "a"
```

4.33 (close <descrfile>)

Questa funzione chiude un file e restituisce NIL. <descrfile> è un descrittore di file ottenuto dal richiamo della funzione OPEN. Dopo CLOSE, il descrittore di file resta invariato ma non è più valido.

Supponendo ad esempio che X è un descrittore di file aperto valido.

```
(close x)
```

chiuderebbe il file associato e restituirebbe NIL.

4.34 (command <argom>...)

Questa funzione esegue comandi AutoCAD dall'interno di AutoLISP e restituisce sempre NIL. Gli argomenti rappresentano i comandi e sottocomandi di AutoCAD e ogni argomento viene valutato e inviato ad AutoCAD in risposta a messaggi di richiesta successivi. I nomi di comandi e di opzioni sono presentati come stringhe, i punti a 2 dimensioni come liste di due numeri reali e i punti a 3 dimensioni come liste di 3 numeri reali. AutoCAD riconosce i nomi dei comandi solo dopo che ha inviato il messaggio "Comando:". Ad esempio:

```
(setq pt1 '(1.45 3.23))
(setq pt2 (getpoint "Immettere un punto: "))
(command "linea" pt1 pt2)
(command "")
```

Supponendo che il messaggio "Comando:" di AutoCAD sia già stato richiamato, questa sequenza di espressioni dà un valore al punto "pt1", sollecita l'immissione del punto "pt2" ed esegue il comando LINEA di AutoCAD con i due punti specificati. Gli argomenti di COMMAND possono essere stringhe, reali, interi o punti a seconda del comando AutoCAD che viene eseguito. Una stringa nulla ("") equivale a digitare uno spazio sulla tastiera. Se si richiama un COMMAND senza specificare argomenti, ciò equivale a premere CTRL C sulla tastiera, cioè ad annullare un comando (vale per la maggior parte dei comandi AutoCAD).

I comandi eseguiti dalla funzione COMMAND non vengono ripetuti sullo schermo se la variabile di sistema CMDECHO di AutoCAD (accessibile tramite SETVAR e GETVAR) ha il valore zero. La funzione COMMAND è il metodo base per accedere a comandi AutoCAD da AutoLISP.

Le funzioni GETxxx (GETANGLE. GETSTRING. GETINT. GETPOINT. ecc.) non possono essere usate nell'ambito della funzione COMMAND. Ogni tentativo in questa direzione produrrà il messaggio "error: AutoCAD rifiuta la funzione". Se viene richiesto un messaggio. bisogna richiamare le funzioni di GET prima. come illustrato sopra. oppure posizionarle tra richiami successivi della funzione COMMAND.

I comandi TESTODIN e SCHIZZO leggono direttamente dalla tastiera e dal digitalizzatore. non possono quindi essere utilizzati con la funzione COMMAND di AutoLISP. COMMAND non può essere utilizzato nemmeno per eseguire PLOT. PLOTST e SCRIPT.

Pausa per immissione dati da parte dell'utente

Se un comando di AutoCAD è in esecuzione e incontra il simbolo predefinito di PAUSA in qualità di argomento della funzione COMMAND, la funzione verrà sospesa per permettere all'utente di immettere dei dati oppure di procedere ad un trascinamento. PAUSA svolge quindi il medesimo ruolo della barra rovesciata per i menù.

Se richiamate un comando trasparente mentre la funzione COMMAND è sospesa, questa rimarrà tale. L'utente potrà quindi eseguire tutti i PAN e gli ZOOM desiderati mentre la funzione permane in pausa. La pausa rimane valida finchè AutoCAD non ottiene l'immissione dei valori richiesti e i comandi trasparenti sono disazionati. Ad esempio:

```
(command "cerchio" "5,5" pause "linea" "5,5" "7,5" "")
```

imposta il comando CERCHIO, assegna al punto centrale le coordinate 5,5 e inserisce una pausa per permettere all'utente di trascinare il raggio del cerchio sullo schermo. Quando l'utente ha selezionato il punto col puntatore oppure ne ha digitato le coordinate, la funzione riprende disegnando una linea da 5,5 a 7,5.

Le selezioni dal menù non vengono sospese dalla pausa in AutoLISP. Se una voce di menù è attivata mentre la funzione di COMMAND è in pausa in attesa dell'immissione dati, i dati possono essere immessi dal menù. Se invece desiderate che anche la voce di menù venga sospesa dovete aggiungerle una barra rovesciata. Dopo che sono stati immessi i dati richiesti sia la funzione COMMAND che la voce di menu torneranno a funzionare regolarmente.

NOTE:

1. Il simbolo di pausa è solitamente costituito da una stringa contenente una barra rovesciata. Potete impostare una barra rovesciata singola. senza servirvi del simbolo di pausa, ma se la funzione di COMMAND è richiamata da una voce di menù, la barra rovesciata otterrà l'effetto di sospendere non la funzione COMMAND ma piuttosto la voce di menù dopo che è stato letta. Il dispositivo di pausa potrà eventualmente aver bisogno di un valore di stimolo differente nelle versioni successive di AutoLISP. Di conseguenza consigliamo di utilizzare il simbolo di PAUSA invece di una semplice barra rovesciata.

2. Se si incontra una pausa mentre un comando è in attesa di una stringa di testo o del valore di un attributo, AutoCAD entrerà in pausa per l'immissione del dato solo se la variabile di sistema TEXTEVAL è impostata con un valore differente da zero. Altrimenti, il valore del simbolo di pausa (la barra rovesciata) verrà preso per il testo e non provocherà nessuna pausa.

3. Quando una funzione di COMMAND è stata sospesa per l'immissione di un dato, tale funzione verrà considerata ancora attiva, in modo che l'utente non potrà richiamare un'altra espressione di AutoLISP per valutarla.

4.35 (cond (<test1> <risultato1>) ...)

Questa funzione accetta una numero qualsiasi di liste come argomenti e valuta il primo elemento di ogni lista (nell'ordine specificato) finché uno di questi elementi restituisce un valore diverso da NIL. A questo punto la funzione valuta quelle espressioni che seguono il test affermativo e restituisce il valore dell'ultima espressione nella sottolista. Se è presente solo un'espressione nella sottolista (se manca <risultato>, ad esempio), viene restituito il valore dell'espressione <test>. COND è la funzione condizionale principale in AutoLISP.

Nell'esempio seguente, COND viene usato per calcolare un valore assoluto.

```
(cond ((minusp a) (- a)
      (t a)
)
```

Se ad "a" è stato dato il valore -10, si ottiene 10. Solitamente T viene usato come espressione <test> standard. Aggiungiamo un ulteriore esempio. Data una stringa di risposta che indichiamo col simbolo "s", questa funzione valuta la risposta e restituisce 1 se tale risposta corrisponde a "Y" o"y" cioè si, e 0 se la risposta è "N" o "n" cioè no, negli altri casi si ottiene NIL.

```
(cond ((= s "Y") 1)
      ((= s "y") 1)
      ((= s "N") 0)
      ((= s "n") 0)
      (t nil)
)
```

4.36 (cons <nuovo primo elemento> <lista>)

Questa è la funzione principale che costruisce liste (CONStructor) ; essa, partendo da un elemento (<nuovo primo elemento>) e una lista (<lista>), restituisce la somma dell'elemento all'inizio della lista. Esempi:

```
(cons 'a '(b c d))       restituisce    (A B C D)
(cons '(a) '(b c d))     restituisce    ((A) B C D)
```

Si noti che il primo elemento può essere un atomo o una lista.

CONS può accettare anche un atomo al posto dell'argomento della <lista>, formando così una struttura conosciuta come *coppia puntata*. Quando visualizza una coppia puntata, AutoLISP pone una virgola o un puntino tra il primo elemento e il secondo. La funzione CDR può essere utilizzata per restituire il secondo atomo di una coppia puntata.
Abbiamo quindi:

```
(cons 'a 'b)             restituisce    (A . B)
(car (cons 'a 'b))       restituisce    A
(cdr (cons 'a 'b))       restituisce    B
```

Una coppia puntata è da considerarsi come un tipo particolare di lista e non viene accettata come argomento di determinate funzioni che gestiscono liste di tipo normale.

4.37 (cos <angolo>)

Questa funzione restituisce il coseno di un <angolo>, in cui <angolo> è espresso in radianti. Esempi:

```
(cos 0.0)      restituisce   1.0
(cos pi)       restituisce  -1.0
```

4.38 (defun <sim> <lista argomenti> <espr>...)

DEFUN definisce una funzione con il nome <sim> (si noti che il nome della funzione viene già restituito letteralmente). Il nome della funzione è seguito da una lista di argomenti (eventualmente vuota) che a sua volta puo essere seguita da una barra obliqua e dai nomi di una o di più simboli locali della funzione. La barra deve essere separata dal primo simbolo locale e dall'ultimo argomento (nei casi in cui ne esiste uno) da almeno uno spazio vuoto. Se non sono presenti ne argomenti nè simboli locali, occorre fornire una serie di parentesi vuote dopo il nome della funzione. Ad esempio:

```
(defun myfunc (x y) ... )       (la funzione dispone di due argomenti)
(defun myfunc (/ a b) ... )     (la funzione dispone di due simboli locali)
(defun myfunc (x / temp) ... )  (un argomento e un simbolo locale)
(defun myfunc () ... )          (nessun argomento o simbolo locale)
```

La lista degli argomenti e delle variabili locali è seguita da una o più espressioni da valutare quando la funzione viene eseguita.

DEFUN restituisce il nome della funzione che è stata definita. Quando la funzione definita è richiamata, i suoi argomenti saranno valutati e legati ai simboli degli argomenti. Le variabili locali possono essere usate all'interno della funzione senza che venga modificato il loro legame ai livelli esterni. La funzione restituirà il risultato dell'ultima espressione valutata. Tutte le espressioni precedenti hanno solo effetti secondari. La funzione DEFUN restituisce il nome della funzione definita. Per esempio:

```
(defun add10 (x)
     (+ 10 x)
)                     restituisce   ADD10
(add10 5)             restituisce   15
(add10 -7.4)          restituisce   2.6
```

e:

```
(defun punti (x y / temp)
     (setq temp (strcat x "..."))
     (strcat temp y)
)                     restituisce   PUNTI
(punti "a" "b")       restituisce   "a...b"
(punti "da" "a")      restituisce   "da...a"
```

Bisogna stare attenti a non usare mai il nome di una funzione incorporata o di una variabile come <sim>, dato che ciò renderebbe la funzione incorporata inaccessibile.

4.38.1 Librerie di funzioni e caricamento automatico

Le definizioni di funzioni possono essere memorizzate in files e caricate mediante la funzione LOAD di AutoLISP, descritta più avanti in questo capitolo. Se esiste il file "acad.lsp", AutoLISP lo caricherà automaticamente ogni volta che si richiama l'Editore di Disegni di AutoCAD; si possono usare queste funzioni per creare una libreria delle funzioni comuni in modo che siano sempre accessibili.

Ogni file di libreria del tipo "lsp." può contenere, oltre a DEFUN, anche altre espressioni. Dal momento che il caricamento di un file induce la valutazione di queste espressioni, potete automatizzare il richiamo di queste funzioni ogni volta che questo file viene caricato. Va notato però che il caricamento di "acad.lsp" avviene prima che l'Editore di disegni di AutoCAD sia pienamente inizializzato, per cui vi sconsigliamo di utilizzare la funzione COMMAND nel file "acad.lsp" al di fuori di un DEFUN. Per fare in modo che il vostro file "acad.lsp" esegua una serie di comandi automaticamente dopo essere stato caricato, includerete un DEFUN della funzione speciale "S::STARTUP". Rimandiamo ai capitoletti seguenti per dettagli.

4.38.2 Funzioni C:XXX -- Aggiungere comandi AutoCAD

Si possono aggiungere nuovi comandi ad AutoCAD mediante DEFUN per definire delle funzioni che implementino questi comandi. Per poter essere usate come comandi in AutoCAD, tali funzioni devono seguire le regole seguenti:

1. La funzione deve avere un nome del tipo "C:XXX", in cui tutte le lettere sono maiuscole. La parte "C:" del nome deve sempre essere presente; la parte "XXX" può essere il nome di un comando a scelta, con la restrizione che non deve essere il nome di un comando già esistente in AutoCAD, compresi i comandi scritti dall'utente.

2. La funzione deve essere definita con una lista di argomenti NIL (ma sono permesse variabili locali).

L'esempio che segue definisce una funzione che disegna un quadrato usando polilinee.

```
(defun C:PQUADRATO (/ pt1 pt2 pt3 pt4 lung)
     (setq pt1 (getpoint "Vertice in basso a sinistra: "))
     (setq lung (getdist pt1 "Lunghezza lato: "))
     (setq pt2 (polar pt1 0.0 lung))
     (setq pt3 (polar pt2 (/ PI 2.0) lung))
     (setq pt4 (polar pt3 PI lung))
     (command "PLINEA" pt1 pt2 pt3 pt4 "C")
     )
```

Le funzioni definite in questo modo possono essere richiamate semplicemente immettendo il nome della funzione quando appare il messaggio "Comando :" di AutoCAD. Se "XXX" non è un comando, AutoCAD prova a richiamare la funzione "C:XXX" di AutoLISP con zero parametri. Per la funzione C:PQUADRATO dell'esempio precedente, il dialogo sarebbe:

Comando: <u>PQUADRATO</u>
Vertice in basso a sinistra: *<u>(immettere un punto)</u>*

Lunghezza lato: *(immettere una distanza)*

In seguito, la funzione richiamerebbe il comando PLINEA di AutoCAD e risponderebbe alle richieste per disegnare il quadrato richiesto.

Aggiungere comandi ad AutoCAD con questo metodo è una caratteristica molto potente di AutoLISP. Il nome del nuovo comando non deve essere scritto fra parentesi quando lo si usa, infatti il comando implementato è trattato come un qualsiasi altro comando di AutoCAD.

4.38.3 Funzioni S::XXX - Esecuzione automatica

Funzioni definite dall'utente i cui nomi iniziano con "S::" verranno richiamate automaticamente quando si presentano determinate situazioni durante una sessione di editazione. Il prefisso "S::" va quindi considerato come "riservato". Per evitare conflitti con funzioni che non hanno niente a che vedere, raccomandiamo di utilizzare questo prefisso per questo tipo di funzioni speciali.

Al momento, l'unica funzione ad esecuzione automatica è "S::STARTUP". Se questa funzione viene definita, verrà richiamata automaticamente (senza argomenti) al momento di entrare nell'Editore di disegni di AutoCAD per creare un nuovo disegno o editare un disegno già esistente. E' possibile naturalmente includere un DEFUN di "S::STARTUP" nel vostro file "acad.lsp" per eseguire operazioni di impostazione prima di cominciare a lavorare con l'editore.

Ad esempio, supponiamo che si desideri sostituire la versione standard dei comandi USCIRE e FINE con una versione personale di tali comandi. E' possibile fare questo con un file "acad.lsp" contenente quanto segue:

```
(defun C:USCIRE ()
    ... la vostra definizione ...
)
(defun C:FINE ()
    ... la vostra definizione ...
)
(defun S:STARTUP ()
      (command "NUOVDEF" "USCIRE")
      (command "NUOVDEF" "FINE")
)
```

4.39 (distance <pt1> <pt2>)

Questa funzione restituisce la distanza tra i punti tridimensionali <pt1> e <pt2>. Esempi:

```
(distance '(1.0 2.5 3.0) '(7.7 2.5 3.0))   restituisce   6.7
(distance '(1.0 2.0 0.5) '(3.0 4.0 0.5))   restituisce   2.82843
```

Se la variabile di sistema FLATLAND è diversa di 0, DISTANCE si aspetta l'immissione di punti bidimensionali (e ignora le componenti Z se vengono forniti punti 3D) e restituisce la distanza bidimensionale tra i punti come se fossero proiettati sul piano di costruzione corrente. Lo stesso si verifica se FLATLAND è diversa da 0 e uno o entrambi i punti forniti sono bidimensionali.

4.40 (eq <espr1> <espr2>)

Questa funzione verifica se <espr1> e <espr2> sono identiche, cioè se sono legate allo stesso oggetto (da SETQ, per esempio). EQ restituisce T se le due espressioni sono identiche e NIL in caso contrario. L'uso tipico di questa funzione è quello di determinare se due liste sono identiche. Determinando per esempio quanto segue:

```
(setq f1 '(a b c))
(setq f2 '(a b c))
(setq f3 f2)
```

allora:

`(equal f1 f3)`	restituisce	NIL	*(f1 e f3 non sono la stessa lista)*
`(equal f3 f2)`	restituisce	T	*(f3 e f2 sono esattamente la stessa lista)*

Vedi anche la funzione EQUAL, qui di seguito.

4.41 (equal <espr1> <espr2>) <approssimazione>

Questa funzione verifica se <espr1> e <espr2> sono identiche, cioè se la loro valutazione dà lo stesso risultato. Determinando per esempio quanto segue:

```
(setq f1 '(a b c))
(setq f2 '(a b c))
(setq f3 f2)
```

allora:

`(eq f1 f3)`	restituisce	T	*(f1 e f3 danno lo stesso risultato)*
`(eq f3 f2)`	restituisce	T	*(f3 e f2 sono esattamente la stessa lista)*

Si noti che due liste identiche (EQUAL) non sono necessariamente EQ, mentre atomi che sono EQUAL, sono sempre anche EQ. Inoltre, due liste o atomi qualsiasi che sono EQ, sono sempre EQUAL.

Nel paragonare due numeri reali (o due liste di numeri reali, come nel caso di punti) è importante tenere presente che due numeri "identici" possono differire leggermente se sono stati utilizzati due metodi differenti per calcolarli. Con l'aiuto dell'argomento numerico opzionale <approssimazione> si può specificare la differenza massima che può intercorrere tra <espr1> e <espr2> senza che queste vengano già considerate differenti. Ad esempio, dati:

```
(setq a 1.123456)
(setq b 1.123457)
```

allora:

`(equal a b)`	restituirebbe	NIL
`(equal a b 0.00001)`	restituirebbe	T

4.42 (eval <es

Restituisce il risul: :ione di <espr> in cui <espr> è un'espressione AutoL!
qualsiasi. Determin: quanto segue:

```
(setq a
(setq b
```

allora:

```
(eval 4.        restituisce   4.0
(eval (a        restituisce   10
(eval a)        restituisce   123
(eval b)        restituisce   123
```

4.43 (exp <nu:

Questa funzione re ɔ all'esponente <numero> (antilogaritmo naturale): restitu
un numero reale. E

```
(exp 1.0        restituisce   2.71828
(exp 2.2        restituisce   9.02501
(exp -0.        restituisce   0.67032
```

4.44 (expt <b: :nte>)

Questa funzione : elevato all'<esponente> specificato. Se entrambi gli argom
sono interi, il risu: In caso contrario il risultato è un numero reale. Esempi:

```
(expt 2         restituisce   16
(expt 3         restituisce   9.0
```

4.45 (findfil :e>)

Questa funzione :grammi utente di cercare all'interno del sentiero di lib
AutoCAD una fi: sentiero di libreria AutoCAD è composto dall'elenco cor
seguito dall'elen: file di disegno corrente, seguito a sua volta dall'e
menzionato dalla :nte ACAD (se specificato) e in ultimo dall'elenco contene
file di programm:

FINDFILE non : :odo da cercare un tipo di file o un'estensione particola:
desiderate un ric :vete fornire esplicitamente il tipo del file da ricercare.
nome non è qua: :one cioè di prefisso di unità disco/elenco). AutoCAD ce
file e ne restitu: :ificato o NIL. se non trova nessun file con questo non
viene fornito un disco/elenco. AutoCAD cerca solo nell'elenco menzionato
esegue quindi u ɔ il sentiero di libreria). Il nome qualificato restitui
FINDFILE può dalla funzione OPEN. (Negli esempi seguenti utilizzian
come separatore : :ell'elenco. in PC-DOS/MS-DOS, si può utilizzare sia "/
"\\".)

Ad esempio, supponiamo che:

- l'elenco corrente è "/acad" e contiene il file "abc.lsp",
- stiamo editando un disegno nell'elenco "/acad/disegni"
- la variabile d'ambiente ACAD contiene l'indicazione "/acad/ausiliario"
- il file "xyz.txt" esiste solo nell'elenco "/acad/ausiliario" e
- il file "nessuno" non è presente in nessun elenco sul sentiero di libreria.

Allora:

```
(findfile "abc.lsp")   restituirebbe   "/acad/abc.lsp"
(findfile "xyz.txt"    restituirebbe   "/acad/ausiliario"xyz.txt"
(findfile "nessuno")   restituirebbe   NIL
```

4.46 (fix <numero>)

Questa funzione restituisce la conversione del <numero> in un intero: il <numero> può essere un intero o un reale. Se è reale, viene approssimato all'intero restante dopo aver eliminato le cifre dopo la virgola. Ad esempio:

```
(fix 3)        restituisce    3
(fix 3.7)      restituisce    3
```

4.47 (float <numero>)

Questa funzione restituisce la conversione del <numero> in un numero reale. Il <numero> può essere intero o reale. Esempi:

```
(float 3)      restituisce    3.0
(float 3.75)   restituisce    3.75
```

4.48 (foreach <nome> <lista> <espr>...)

Questa funzione percorre la <lista> assegnando un <nome> ad ogni elemento e valutando ogni <espr> per ogni elemento nella lista. Si puo specificare un numero qualsiasi di espressioni. FOREACH restituisce il risultato dell'ultima <espr> valutata. L'esempio:

```
(foreach n '(a b c) (print n))
```

è equivalente a:

```
(print a)
(print b)
(print c)
```

solo che FOREACH restituisce unicamente il risultato della valutazione dell'ultima espressione.

4.49 (gcd <num1> <num2>)

Questa funzione restituisce il maggiore denominatore comune di <num1> e <num2>, che devono essere interi: Esempi:

```
(gcd 81 57)        restituisce   3
(gcd 12 20)        restituisce   4
```

4.50 (getangle [<pt>] [<richiesta>])

Questa funzione introduce una pausa per permettere all'utente di immettere un angolo. <richiesta> è una stringa opzionale che viene visualizzata come messaggio di richiesta e <pt> è un punto opzionale di base 2D nell'UCS corrente. L'angolo restituito è espresso in radianti ed è relativo al piano di costruzione corrente (il piano XY dell'UCS corrente all'elevazione corrente).

E' possibile specificare un angolo digitando un numero nel formato di unità di misura per angoli corrente in AutoCAD; comunque, anche se il formato corrente di unità angolari è in gradi sessagesimali o centesimali, questa funzione restituisce sempre un angolo in radianti.

Si può anche "indicare" ad AutoLISP l'angolo desiderato puntando su due posizioni bidimensionali dello schermo grafico. Durante questa operazione AutoCAD visualizza un cursore a linea elastica dal primo punto alla posizione corrente del puntatore a croce, per facilitare la scelta del secondo punto. L'argomento opzionale di GETANGLE, <pt>, viene interpretato (se specificato) come primo dei due punti da specificare. E' possibile specificare un punto di base 3D ma potrebbe creare confusione dal momento che l'angolo viene sempre misurato nel piano di costruzione corrente. Esempi:

```
(setq ang (getangle))
(setq ang (getangle '(1.0 3.5))
(setq ang (getangle "In quale direzione? ")
(setq ang (getangle '(1.0 3.5) "In quale direzione? ")
```

Non si può immettere un'altra espressione LISP in risposta a una richiesta di GETANGLE. Se lo si facesse, si otterrebbe il messaggio "Can't reenter AutoLISP" (impossibile rientrare in AutoLISP). Per ulteriori informazioni vedi GETORIENT e INITGET.

4.51 (getcorner <punto> [<messaggio>])

La nuova funzione getcorner restituisce un punto nell'UCS corrente, esattamente come GETPOINT. l'unica differenza è che GETCORNER esige un argomento di base del punto <pt> e disegna un rettangolo partendo da questo punto seguendo il movimento sullo schermo che viene fatto con l'aiuto del puntatore. Per ulteriori informazioni vedi GETORIENT e INITGET.

Il punto di base è espresso relativamente all'UCS corrente. Se viene fornito un punto 3D, la sua componente Z viene ignorata e l'elevazione corrente viene utilizzata come valore per la Z.

Non è possibile immettere nessun'altra espressione di AutoLISP in risposta alla richiesta di GETCORNER.

4.52 (getdist [<pt>] [<richiesta>])

Questa funzione introduce una pausa per permettere all'utente di immettere una distanza. <richiesta> è una stringa opzionale che viene visualizzata come messaggio di richiesta e <pt> è un punto 2D o 3D di base opzionale appartenente all'UCS corrente. E' possibile specificare una distanza digitando un numero nel formato di unità di misura per distanze corrente di AutoCAD: comunque la funzione restituisce sempre una distanza come numero reale.

Si puo anche indicare ad AutoLISP la distanza desiderata puntando su due posizioni dello schermo grafico. Durante questa operazione AutoCAD visualizza un cursore a linea elastica dal primo punto alla posizione corrente del puntatore a croce, per facilitare la scelta del secondo punto. L'argomento opzionale di GETDIST, <pt>, viene interpretato (se specificato) come primo dei due punti da specificare.

Se utilizzate il metodo dei due punti, GETDIST controlla la variabile di sistema FLATLAND e l'identificatore di controllo per punto tridimensionale della funzione INITGET per determinare la maniera in cui deve calcolare la distanza. Se FLATLAND è 0, restituisce la distanza 3D tra i due punti. Se FLATLAND è diverso da 0, restituisce una distanza 3D solo se l'identificatore di punto 3D di INITGET è impostato, altrimenti si comporta come se fossero stati forniti punti 2D.

Esempi:

```
(setq dist (getdist))
(setq dist (getdist '(1.0 3.5))
(setq dist (getdist "Lungo quanto? ")
(setq dist (getdist '(1.0 3.5) "Lungo quanto? ")
```

Non si puo immettere un'altra espressione LISP in risposta a una richiesta di GETDIST. Rinviamo a INITGET.

4.53 (getenv <nome della variabile>)

Questa funzione restituisce il valore di stringa assegnato a una variabile d'ambiente. L'argomento <nome della variabile> è una stringa che specifica il nome della variabile che deve essere letta. Se questa variabile non esiste, AutoLISP restituisce NIL.

Ad esempio, se la variabile d'ambiente ACAD contiene "/acad/ausiliario/" e non esiste nessuna variabile che sia chiama NESSUNO, allora:

```
(getenv "ACAD")       restituirebbe    "/acad/ausiliario/"
(getenv "NESSUNO")    restituirebbe    NIL
```

Va notato che nei sistemi basati su UNIX, "ACAD" e "acad" vengono considerate come due variabili differenti, dato che UNIX distingue tra maiuscole e minuscole.

4.54 (getint [<richiesta>])

Questa funzione introduce una pausa per permettere all'utente di immettere un intero e restituisce quell'intero. I valori possono essere compresi nell'intervallo da -32768 a +32767. <richiesta> è una stringa opzionale che viene visualizzata come messaggio di richiesta. Ad esempio:

```
(setq numero (getint))
(setq numero (getint "Immettere un numero: "))
```

Non si può immettere un'altra espressione LISP in risposta a una richiesta di GETINT. Rinviamo a INITGET.

4.55 (getkword [<messaggio>])

La funzione GETKWORD richiede da parte dell'utente l'immissione di una parola chiave. Utilizzando la funzione INITGET (descritta più sotto), si ottiene una lista delle parole chiave valide. GETKWORD restituisce la parola chiave corrispondente a quanto immesso dall'utente. trasformandola in una stringa di caratteri. AutoCAD riproverà se quanto immesso non corrisponde a una delle parole chiave. Un'immissione nulla (se permessa) provoca la risposta NIL. Si ottiene NIL anche quando non si è stabilita nessuna stringa contenente una parola chiave. Ad esempio:

```
(initget 1 "Si No")
(setq x (getword "Sei sicuro? (Si o No)"))
```

chiederà all'utente di rispondere Si oppure No e porrà una X in corrispondenza della risposta ottenuta. Se la risposta non corrisponde a nessuna delle parole chiave, AutoCAD chiederà all'utente di riprovare.

Non si può immettere un'altra espressione di LISP in risposta ad una richiesta di GETKWORD. Rinviamo a INITGET.

4.56 (getorient [<pt>] [<messaggio>])

In AutoLISP gli angoli sono sempre rappresentati in radianti, con la direzione 0 volta verso destra e gli angoli crescenti in senso trigonometrico. Nei casi in cui l'utente ha scelto una direzione differente per il radiante 0 o per la crescita degli angoli si rivelano necessarie alcune riconversioni, realizzabili tramite il comando UNITA o le variabili di sistema ANGBASE e ANGDIR.

La funzione GETORIENT è simile a GETANGLE ma, per quanto riguarda la base di 0 gradi e la direzione di crescita dell'angolo, si comporta diversamente da GETANGLE. GETANGLE dovrebbe essere usata quando si vuole sapere il valore di una rotazione (angolo relativo), mentre GETORIENT va usata se si vuole sapere l'orientamento (angolo assoluto).

Supponiamo che il comando UNITA sia stato impiegato per selezionare un angolo di 90 gradi (a nord) e il senso orario come direzione di crescita dell'angolo. La tabella seguente mostra quali valori (in radianti) GETORIENT e GETANGLE emetteranno come valori rappresentativi per l'input (in gradi) da parte dell'utente.

Input(gradi)	GETANGLE	GETORIENT
0	0.0	1.5708
-90	1.5708	3.14159
180	3.14159	4.71239
90	4.71239	0.0

Come mostra la tabella, GETANGLE rispetta la direzione dell'angolo, ma ignora la direzione di 0 gradi. E' naturalmente possibile impiegare GETANGLE per ottenere un'ampiezza di rotazione per l'inserimento di un blocco, dal momento che 0 gradi corrispondono sempre a 0 radianti. GETORIENT, invece, rispetta sia la direzione di 0 gradi che il senso di crescita dell'angolo. Si può quindi usare GETORIENT per ottenere angoli come quello della linea di base per una voce di testo. Ad esempio, se tramite il comando UNITA si è selezionato un angolo di 90 gradi, la linea di base secondo cui si orienterà una normale linea di testo orizzontale sarà di 90 gradi.

Come per GETANGLE, l'angolo restituito sarà espresso in radianti e sarà relativo al piano di costruzione corrente.

Non è possibile immettere un'altra funzione di LISP in risposta ad una richiesta di GETORIENT. Rinviamo a GETANGLE e a INITGET.

4.57 (getpoint [<pt>] [<richiesta>])

Questa funzione introduce una pausa per permettere all'utente di immettere un punto. <pt> è un punto 2D o 3D opzionale di base relativo all'UCS corrente e <richiesta> è una stringa opzionale che viene visualizzata come messaggio di richiesta. E' possibile specificare un punto mediante puntamento o digitandone le coordinate nel formato di unità di misura corrente. Se è presente l'argomento opzionale <pt>, AutoCAD visualizza un cursore a linea elastica da quel punto alla posizione corrente del puntatore a croce. GETPOINT restituisce un punto, cioè una lista di due numeri reali. Esempi:

```
(setq p (getpoint))
(setq P (getpoint "Dove?"))
(setq p (getpoint '(1.5 2.0) "Secondo punto: "))
```

Il punto restituito è espresso relativamente all'UCS corrente. Se la variabile di sistema FLATLAND è 0, GETPOINT restituisce un punto 3D, altrimenti viene restituito un punto 2D a meno che INITGET non sia stata utilizzata per apporre un identificatore di controllo per punto tridimensionale, in tal caso il punto restituito sarà 3D.

Non si può immettere un'altra espressione LISP in risposta a una richiesta di GETPOINT. Rinviamo a GETCORNER e INITGET.

4.58 (getreal [<richiesta>])

Questa funzione introduce una pausa per permettere all'utente di immettere un numero reale e restituisce quel numero reale. <richiesta> è una stringa opzionale che viene visualizzata come messaggio di richiesta. Esempi:

```
(setq val (getreal))
(setq val (getreal "Fattore di scala: "))
```

Non si può immettere un'altra espressione LISP in risposta a una richiesta di GETREAL. Rinviamo a INITGET.

4.59 (getstring [<cr>] [<richiesta>])

GETSTRING introduce una pausa per permettere all'utente di immettere una stringa e restituisce quella stringa. Se la stringa supera i 132 caratteri, solo i primi 132 caratteri verranno restituiti. Se <cr> è presente ed è diverso da NIL, la stringa immessa può contenere spazi vuoti (e deve quindi essere conclusa da RETURN). <richiesta> è una stringa opzionale che viene visualizzata come messaggio di richiesta. Esempi:

```
(setq s (getstring))
(setq s (getstring "Cognome? "))
(setq s (getsring T "Nome e cognome? "))
```

Se l'input dell'utente corrisponde ad una delle opzioni (parole chiave), è possibile utilizzare la funzione GETKWORD, descritta più sopra, al posto di GETSTRING.

Non si può immettere un'altra espressione LISP in risposta a una richiesta di GETSTRING.

4.60 (getvar <nome variabile>)

Questa funzione restituisce il valore di una variabile di sistema di AutoCAD. Il nome della variabile deve essere scritto in maiuscolo e tra virgolette. Supponendo ad esempio che l'ultimo raggio di raccordo specificato sia di 0.25 unità:

```
(getvar "FILLETRAD")        restituirebbe 0.25
```

Se cercate di utilizzare GETVAR per restituire il valore di una variabile di sistema sconosciuta ad AutoCAD, otterrete NIL. Rimandiamo all'Appendice A della Guida all'Uso di AutoCAD per l'elenco delle variabili di sistema correnti di AutoCAD. Vedi anche la funzione SETVAR.

4.61 (graphscr)

La funzione GRAPHSCR cambia dallo schermo testo a quello grafico su sistemi monoschermo (stesso effetto del tasto "FLIP SCREEN" di AutoCAD). GRAPHSCR restituisce sempre NIL. Vedi anche la funzione TEXTSCR.

4.62 (if <espr test> <espr then> [<espr else>])

Questa funzione valuta espressioni in modo condizionale: se <espr test> è diversa da NIL, valuta l'<espr then>. In caso contrario valuta l'<espr else>, che è opzionale. IF restituisce il valore dell'espressione selezionata; se <espr else> manca e <espr test> è diversa da NIL, IF restituisce NIL. Esempi:

```
(if (= 1 3) "SI!!" "no.")          restituisce    "no."
(if (= 2 (+ 1 1)) "SI!!")          restituisce    "SI!!"
(if (= 2 (+ 3 4)) "SI!!")          restituisce    NIL
```

4.63 (initget [<bits>] [<stringa>])

Questa nuova funzione mette a disposizione varie opzioni impiegabili con la funzione di GET chiamata in seguito (eccezion fatta per le funzioni GETSTRING e GETVAR). INITGET restituisce sempre NIL. L'argomento opzionale <bits> è un numero intero con questi valori:

Bits di INITGET	Significato
1	Respinge ogni input nullo
2	Respinge tutti i valori uguali a 0
4	Respinge tutti i valori negativi
8	Non controlla i limiti, anche quando LIMCHECK è inserito
16	Restituisce punti tridimensionali invece che bidimensionali
32	Utilizza linee tratteggiate per disegnare la linea elastica e i riquadri di selezione

I bits possono essere combinati tra loro in tutte le maniere possibili per formare un valore tra 0 e 63. Nelle versioni future di AutoLISP potranno essere inseriti ulteriori bits di controllo INITGET, è perciò consigliabile evitare l'uso di bits non documentati. Se la variabile di sistema FLATLAND è 0 si suppone che l'identificatore di controllo per punto tridimensionale sia stato impostato.

Se l'immissione dell'utente non corrisponde ad una o più condizioni postulate (ad esempio viene immesso un valore 0 quando si è stabilito che i valori 0 non sono validi), AutoCAD visualizzerà un messaggio e chiederà all'utente di riprovare. Ad esempio:

```
(initget (+ 1 2 4))
(setq age (getint "Quanti anni hai? "))
```

se viene inserito un valore nullo, negativo o pari a 0, l'utente riceverà immediatamente il messaggio che gli chiede di immettere altri valori. Se non viene fornito nessun argomento <bits>, 0 (nessuna condizione) viene assunto come valore. I valori di controllo speciali sono accettati solo da quelle funzioni di GET per le quali hanno senso, come vediamo nella tabella seguente.

La valore di controllo 32 viene accettato dalle funzioni GETPOINT, GETCORNER, GETDIST, GETANGLE e GETORIENT quando viene fornito un punto di base e le induce ad utilizzare linee tratteggiate (o evidenziate in una qualunque altra forma) per visualizzare la linea elastica

o il cursore a riquadro che partono dal punto base indicato. Se la variabile di sistema POPUPS corrisponde a zero, indica cioè che il programma di gestione dei dispositivi di visualizzazione non gestisce l'interfaccia utente avanzata, AutoCAD ignorerà questo valore INITGET.

Funzione	I bits di controllo INITGET (* sta per accettati)					
	1	2	4	8	16	32
GETINT	*	*	*			
GETREAL	*	*	*			
GETDIST	*	*	*		*	*
GETANGLE	*	*				*
GETORIENT	*	*				*
GETPOINT	*			*	*	*
GETCORNER	*			*	*	*
GETKWORD	*					
GETSTRING						
GETVAR						

L'argomento <stringa> opzionale di INITGET definisce una lista di parole chiave a scelta da verificarsi appena viene chiamata una funzione di GET se l'utente non esegue l'immissione prestabilita (es. non inserisce un punto con la funzione GETPOINT). Se quanto immesso dall'utente corrisponde ad una delle parole chiave della lista, la parola chiave selezionata viene restituita dalla funzione GET in azione come risultato e sotto forma di stringa. Il programma utente è in grado di controllare ogni singola parola chiave e eseguire l'azione richiesta da ciascuna. Se l'immissione dell'utente non è del tipo prestabilito e non corrisponde a nessuna parola chiave, AutoCAD chiederà di riprovare.

La lista delle parole chiave deve essere del tipo "Chiave1 Chiave2 Chiave3, ABBREV3". Le singole parole chiave vengono separate da spazi. Abbreviazioni sono opzionali e possono avvenire in due modi diversi. La porzione richiesta può essere in lettere maiuscole e il resto in minuscolo, oppure la porzione richiesta può essere ripetuta e separata dalla parola chiave con una virgola. La seconda maniera è adatta a favorire applicazioni realizzate in lingue straniere, nelle quali la riconversione minuscolo-maiuscolo puo essere difficoltosa o impossibile. Con entrambi i metodi, l'abbreviazione deve essere inserita correttamente e in tutta la sua lunghezza. Ad esempio:

```
"TIPOLINEA,TI"     e
"TIpolinea"
```

sono specificazioni equivalenti. Entrambi indicano che l'immissione da parte dell'utente di "TIPOLINEA", "TIPO", "TIP" e "TI" è considerata accettabile, "T" però non è sufficiente e "TIPOSNAP" e "TIPREF" non corrispondono a quanto desiderato.

Consideriamo la seguente funzione applicativa:

```
(defun getnum ( / x)
   (initget 1 "Pi Due-Pi")
   (setq x (getreal "Pi/Due-pi/<numero>: "))
   (cond ((eq x "Pi")       pi)
         ((eq x "Due-Pi")  (* 2.0 pi))
         (T                 x)
   )
)
```

Questa funzione INITGET vieta ogni immissione nulla e stabilisce una lista di 2 parole chiave. "Pi" e "Due Pi". GETREAL viene quindi usato per ottenere un numero reale in risposta al messaggio "Pi/Due-pi/<numero>:", il risultato è posto nel simbolo locale X. Se l'utente immette un numero, questo numero viene restituito dalla funzione GETNUM. Però, se l'utente inserisce il numero chiave "Pi" (o semplicemente "P"), GETREAL restituirà la parola chiave "Pi". La funzione COND si accorge di questo e restituisce il valore di *pi*. Con la parola chiave "Due-Pi" succede la medesima cosa.

Gli identificatori e la lista delle parole chiave stabiliti da INITGET si applicano alle chiamate seguenti di tipo GET e vengono successivamente abbandonati. Questo evita all'utente di dover eliminare queste condizioni speciali quando chiama un'altra funzione.

4.64 (inters <pt1> <pt2> <pt3> <pt4> [<sulseg>])

La funzione INTERS esamina due linee e restituisce il punto in cui intersecano o NIL se non intersecano. <pt1> <pt2> sono le estremità della prima linea e <pt3> <pt4> sono le estremità della seconda linea. Tutti i punti sono espressi relativamente all'UCS corrente. Se la variabile di sistema FLATLAND è 0 e tutti e quattro gli argomenti di punto sono 3D, INTERS controlla se esistono intersezioni 3D, se non ne trova, INTERS proietta le linee sul piano di costruzione corrente e cerca solo le intersezioni in 2D.

Se l'argomento opzionale <sulseg> è presente ed è NIL, le linee verranno considerate di lunghezza infinita e INTERS restituirà il punto in cui intersecano, anche se questo si trova oltre l'estremità di una o di entrambe le linee. Se l'argomento opzionale <sulseg> non è presente o non è NIL, il punto d'intersezione deve trovarsi su entrambe le linee, in caso contrario INTERS restituirà NIL. Ad esempio, se:

```
(setq a '(1.0 1.0) b '(9.0 9.0))
(setq c '(4.0 1.0) d '(4.0 2.0))
```

allora:

```
(inters a b c d)          restituisce   NIL
(inters a b c d T)        restituisce   NIL
(inters a b c d NIL)      restituisce   (4.0 4.0)
```

4.65 (itoa <int>)

Questa funzione restituisce la conversione di un intero in una stringa. Esempio:

```
(itoa 33)        restituisce    "33"
(itoa -17)       restituisce    "-17"
```

4.66 (lambda <argomenti> <espr>...)

LAMBDA definisce una funzione "anonima" che viene utilizzata soprattutto quando i tempi spesi dal sistema operativo non giustificano la definizione di una nuova funzione. LAMBDA restituisce il valore della sua ultima <espr> e viene usata spesso in congiunzione con APPLY e/o MAPCAR per eseguire una funzione su una lista. Ad esempio:

```
(apply '(lambda (x y z)
                (* x (- y z))
        )
        '(5 20 14)
)                          restituisce    30
```

e:

```
(setq counter 0)
(mapcar '(lambda (x)
                (setq counter (1+ counter))
                (* x 5)
         )
         '(2 4 -6 10.2)
)                          restituisce    (10 20 -30 51.0)
```

4.67 (last <lista>)

Questa funzione restituisce l'ultimo elemento della <lista>. <lista> non può essere NIL. Esempi:

```
(last '(a b c d e))       restituisce    E
(last '(a b c (d e)))     restituisce    (D E)
```

LAST puo restituire un atomo o una lista.

A prima vista LAST potrebbe sembrare la maniera ideale per ottenere la coordinata Y di un punto. Questo vale per i punti bidimensionali (lista di due numeri reali), mentre nel caso di punti tridimensionali, LAST restituirà la componente Z del punto. Consigliamo quindi, per evitare confusioni, di utilizzare per i punti bidimensionali la funzione CADR per ottenere la Y e, per i punti tridimensionali, CADDR per ottenere la Z.

4.68 (length <lista>)

Questa funzione restituisce un intero che indica il numero di elementi nella <lista>. Esempi:

```
(length '(a b c d))        restituisce   4
(length '(a b (c d)))      restituisce   3
(length '())               restituisce   0
```

4.69 (list <espr>...)

Questa funzione restituisce una lista i cui elementi sono i suoi argomenti. Esempi:

```
(list 'a 'b 'c)            restituisce   (A B C)
(list 'a '(b c) 'd)        restituisce   (A (B C) D)
(list 3.9 6.7)             restituisce   (3.9 6.7)
```

In AutoLISP, questa funzione e usata sovente per definire una variabile di punto bidimensionale o tridimensionale (lista di due o tre numeri reali).

4.70 (listp <elemento>)

Questa funzione restituisce T se <elemento> è una lista e NIL in caso contrario. Ad esempio:

```
(listp '(a b c))      restituisce   T
(listp 'a)            restituisce   NIL
(listp 4.343)         restituisce   NIL
```

4.71 (load <nome file>) [<per difetto>])

Questa funzione carica un file di espressioni AutoLISP e le valuta. <nome file> è una stringa che rappresenta il nome del file senza estensione (".lsp" viene aggiunta automaticamente). <nome file> può contenere un prefisso d'elenco, come in "/funzione/test1". Sui sistemi MS-DOS/PC-DOS si può digitare anche una lettera indicante l'unità disco e usare la barra rovesciata "\" (ma bisogna tenere presente che per ottenere una barra rovesciata in una stringa occorre digitare "\\".)

Se non viene incluso il prefisso dell'elenco nella stringa <nome file>, LOAD ispeziona il sentiero di libreria di AutoCAD alla ricerca del file specificato in una maniera molto simile alla funzione FINDFILE. Se il file viene trovato, LOAD lo carica.

Se l'operazione riesce, LOAD restituisce il valore dell'ultima espressione nel file, che nella maggior parte dei casi, corrisponderà al nome dell'ultima funzione definita nel file. Se l'operazione fallisce, si ottiene di regola un errore AutoLISP. Se però viene fornito l'argomento <per difetto> LOAD restituisce il valore di questo argomento al posto che produrre l'errore. Questo permette ad un'applicazione di AutoLISP che richiama LOAD di eseguire un'operazione alternativa se non viene trovato il file da caricare. Naturalmente occorre accertarsi che l'argomento <per difetto> sia differente dall'ultima espressione nel file, altrimenti il significato del valore restituito da LOAD sarà ambiguo.

Supponendo ad esempio che il file "/fred/test1.lsp" contenga:

```
(defun MIA-FUNZ1 (x)
    ... funzione corpo ...
)
(defun MIA-FUNZ2 (x)
    ... funzione corpo ...
)
```

e che il file "test2.lsp" non esista, allora:

```
(load "/fred/test1")              restituirebbe  MIA-FUNZ2
(load "/fred/test1" "male")       restituirebbe  MIA-FUNZ2
(load "test2" "male")             restituirebbe  "male"
(load "test2")                provocherebbe un errore AutoLISP
```

La funzione LOAD puo essere richiamata dall'interno di un'altra funzione LISP e perfino ricorsivamente (dall'interno del file da caricarsi).

Ogni volta che inizia una sessione nell'Editore di Disegni di AutoCAD, AutoLISP carica il file "acad.lsp", se esiste. In questo file si possono inserire definizioni di funzioni e comandi ricorrenti, che verranno valutati automaticamente ogni volta che si inizia ad editare un disegno. Se desiderate che una serie di comandi AutoCAD o di funzioni AutoLISP vengano eseguite automaticamente all'inizio di una sessione d'editazione, posizionate un DEFUN della funzione speciale "S::STARTUP" nel file "acad.lsp"; se questa funzione esiste, AutoCAD la esegue automaticamente all'inizio della sessione d'editazione. (Rimandiamo alla descrizione di DEFUN per l'esempio esplicativo)

4.72 (log <numero>)

Questa funzione restituisce il logaritmo naturale del <numero> sottoforma di numero reale. Esempi:

```
(log 4.5)      restituisce    1.50408
(log 1.22)     restituisce    0.198851
```

4.73 (logand <numero> <numero>...)

Questa funzione restituisce il risultato di un AND logico di una lista di <numeri>. I numeri devono essere interi. Il risultato sarà un intero. Esempi:

```
(logand 7 15 3)       restituisce   3
(logand 2 3 15)       restituisce   2
(logand 8 3 4)        restituisce   0
```

4.74 (logior <numero> <numero>...)

Questa funzione restituisce il risultato di un OR comprensivo logico di una lista di <numeri>. I numeri devono essere interi. Il risultato sarà un intero. Esempi:

```
(logior 1 2 4)      restituisce   7
(logior 9 3)        restituisce   11
```

4.75 (lsh <num1> <num bit>)

Questa funzione restituisce lo spostamento logico di <num1> per il numero di bit definito da <num bit>. <num1> e <num bit> devono essere interi. Il risultato sara un intero.

Se <num bit> è positivo, <num1> viene spostato verso sinsitra; se è negativo, verso destra. In entrambi i casi "zero" bit vengono spostati verso l'interno e i bit spostati all'esterno sono eliminati. Se un bit "1" è spostato dentro o fuori dal primo bit (il 16esimo su computer sotto DOS, il 32esimo su stazioni di lavoro a 32-bit), il suo segno cambia.

```
(lsh 2 1)          restituisce   4
(lsh 2 -1)         restituisce   1
(lsh 40 2)         restituisce   160
(lsh 16384 1)      restituisce   -32768 su computer sotto DOS
(lsh 12384 1)      restituisce   32768 su stazioni di lavoro a 32-bit
```

4.76 (mapcar <funzione> <lista1> ... <listan>)

MAPCAR restituisce il risultato dell'esecuzione della <funzione> con gli elementi singoli di <lista1> avanzando lungo la <listan> che sono fornite come argomenti di <funzione>. Il numero di liste deve corrispondere al numero di argomenti richiesti dalla <funzione>. Ad esempio:

```
(setq a 10 b 20 c 30)
(mapcar '1+ '(10 20 30))      restituisce     (11 21 31)
```

Che è equivalente a:

```
(1+ a)
(1+ b)
(1+ c)
```

solo che MAPCAR restituisce una lista dei risultati. In modo analogo:

```
(mapcar '+ '(10 20 30) '(4 3 2))    restituisce    (14 23 32)
```

Che è equivalente a:

```
(+ 10 4)
(+ 20 3)
(+ 30 2)
```

La funzione LAMBDA può specificare una funzione "anonima" che viene eseguita da MAPCAR. Ciò è utile quando alcuni degli argomenti della funzione sono costanti o sono forniti in altro modo. Esempio:

```
(mapcar '(lambda (x) (+ x 3)) '(10 20 30))   restituisce   (13 23 33)
```

e:

```
(mapcar '(lambda (x y z)
                (* x (- y z))
         )
      '(5 6)  '(20 30)  '(14 5.0)
)                            restituisce     (30 150.0)
```

4.77 (max <numero> <numero>...)

Questa funzione restituisce il maggiore dei <numeri> dati. Ogni <numero> può essere un numero reale o intero. Esempi:

```
(max 4.07 -144)            restituisce     4.07
(max -88 19 5 2)           restituisce     19
```

4.78 (member <espr> <lista>)

Questa funzione ricerca la <lista> e se incontra una <espr> restituisce il resto della <lista> iniziando con la prima <espr>. Se non vi sono <espr> nella <lista>, MEMBER restituisce NIL. Esempi:

```
(member 'c '(a b c d e))       restituisce     (C D E)
(member 'q '(a b c d e))       restituisce     NIL
```

4.79 (menucmd <stringa>)

La funzione MENUCMD permette ai programmi LISP di cambiare da un sottomenù all'altro di un menù AutoCAD. Così un programma LISP è in grado di operare con un file di menù associato, visualizzando un sottomenù appropriato di altenative ogni volta che vengono richiesti dati dall'utente. MENUCMD restituisce sempre NIL. L'argomento <stringa> di MENUCMD si presenta di questo tipo:

sezione=sottomenù

in cui:

sezione — Specifica la sezione del menù. I nomi permessi sono:

S	per il menù SCREEN (di schermo)
B	per il menù BUTTONS (dei pulsanti)
I	per il menù ICON
P1 - P10	per i menu a rotolo (POP) da 1 a 10
T1 - T4	per i menù TABLET da 1 a 4 (di tavoletta)
A1	per il menu AUX1 (della tastiera ausiliaria)

sottomenù — specifica quale sottomenu deve essere attivato. Il nome deve essere uno dei titoli di sottomenu (senza gli asterischi "**") che si trova nel file di menù richiamato in memoria in quel momento, oppure il nome di una sezione, come definito sopra.

Rimandiamo all'Appendice B della Guida all'Uso di AutoCAD per ulteriori informazioni. Occorre notare che il prefisso "S", che viene usato per individuare i sottomenu in un file di menu, non appare qui. Per esempio:

```
(menucmd "S=OSNAP")
```

richiamerebbe il sottomenu di "OSNAP" sullo schermo (a condizione che questo sottomenu esista nel file di menu corrente). In modo analogo:

```
(menucmd "B="MIO-PULSANTI")
```

assegnerebbe il sottomenu "MIO-PULSANTI" ai menu dei pulsanti.

Per i menu ad icone e a rotolo, "*" è un nome valido per un sottomenu e produce la visualizzazione del sottomenu corrispondente alla sezione di menu specificata. Ad esempio, questa sequenza:

```
(menucmd "P1=NUMERIC")
(menucmd "P1=*")
```

farà corrispondere al menu a rotolo 1, il sottomenu "NUMERICO" e lo farà apparire sullo schermo.

4.80 (min <numero> <numero>...)

Questa funzione restituisce il più piccolo dei <numeri>. Ogni <numero> può essere reale o intero. Esempi:

(min 683 -10.0)	restituisce	-10.0
(min 73 2 48 5)	restituisce	2

4.81 (minusp <elemento>)

Questa funzione restituisce T se <elemento> è un numero reale o intero, in caso contrario restituisce NIL. Questa funzione non è definita per altri tipi di <elemento>. Esempi:

(minusp -1)	restituisce	T
(minusp -4.293)	restituisce	T
(minusp 830.2)	restituisce	NIL

4.82 (not <elemento>)

Questa funzione restituisce T se l'espressione dà NIL e NIL in caso contrario. Di solito la funzione NULL è usata per liste e la funzione NOT per altri tipi di dati in congiunzione con alcune funzioni di controllo. Determinando per esempio quanto segue:

```
(setq a 123)
(setq b "stringa")
(setq c NIL)
```

allora:

```
(not a)        restituisce   NIL
(not b)        restituisce   NIL
(not c)        restituisce   T
(not '())      restituisce   T
```

4.83 (nth <n> <lista>)

Questa funzione restituisce l'ennesimo elemento di una <lista>, in cui <n> è il numero dell'elemento da restituire (zero è il primo elemento). Se <n> è maggiore del numero di elementi nella <lista>, viene restituito NIL. Esempi:

```
(nth 3 '(a b c d e))     restituisce   D
(nth 0 '(a b c d e))     restituisce   A
(nth 5 '(a b c d e))     restituisce   NIL
```

4.84 (null <elemento>)

Questa funzione restituisce T quando l'elemento è legato a NIL e NIL in caso contrario. Determinando per esempio quanto segue:

```
(setq a 123)
(setq b "stringa")
(setq c NIL)
```

allora:

```
(null a)       restituisce   NIL
(null b)       restituisce   NIL
(null c)       restituisce   T
(null '())     restituisce   T
```

4.85 (numberp <elemento>)

Questa funzione restituisce T se l'<elemento> è un numero reale o intero e NIL in caso contrario. Determinando per esempio quanto segue:

```
(setq a 123)
(setq b 'a)
```

allora:

```
(numberp 4)              restituisce   T
(numberp 3.8348)         restituisce   T
(numberp "Salve")        restituisce   NIL
(numberp 'a)             restituisce   NIL
(numberp a)              restituisce   T
(numberp b)              restituisce   NIL
(numberp (eval b))       restituisce   T
```

4.86 (open <nome file> <modo>)

Questa funzione apre un file di accesso per le funzioni I/O di AutoLISP. Essa restituisce un descrittore di file che deve essere usato da altre funzioni I/O; per questa ragione le deve essere assegnata una variabile tramite SETQ. Esempio:

```
(setq a (open "file.estens" "r"))
```

Il <nome file> è una stringa di testo che specifica il nome e l'estensione del file da aprire. <modo> è l'identificatore "read/write" e deve essere una stringa contenente una sola lettera in minuscolo. Riportiamo qui sotto le diverse lettere che fungono da identificatore.

Modo di OPEN	Descrizione
"r"	File aperto per la lettura, se <nome file> non esiste, la funzione restituisce NIL.
"w"	File aperto per la scrittura. Se <nome file> non esiste, un nuovo file viene creato e aperto. Se invece <nome file> già esiste, i nuovi dati inseriti vengono sovrapposti a quelli che vi sono contenuti.
"a"	File aperto per l'aggiunta di ulteriori dati. Se <nome file> non esiste, un nuovo file viene creato e aperto. Se invece <nome file> già esiste, il file viene aperto nel punto dove terminano i dati, in modo che ogni nuovo dato venga scritto nel file, verrà aggiunto ai dati già esistenti. Nei sistemi DOS, alcuni programmi ed editori di testi scrivono i files con un contrassegno alla fine del testo (CTRL Z, codice ASCII 26 decimale) che indica che il file è terminato. Quando legge un file di testo, DOS considera il file terminato ogni volta che incontra uno di questi contrassegni, indipendentemente dal fatto se dopo di questo si trovano ancora dei dati. Se si intende usare il modo "a" del comando OPEN per aggiungere dati a file prodotti da un'altro programma, bisognerebbe accertarsi che il nuovo programma non inserisca automaticamente contrassegni del tipo CTRL Z alla fine dei file di testo.

Supponendo che negli esempi che seguono non esistano i nomi dei file:

```
(setq f (open "nuovo.tst" "w"))      restituisce   <File #nnn>
(setq f (open "nessun.fil" "r"))     restituisce   NIL
(setq f (open "logfile" "a"))        restituisce   <File #nnn>
```

<nome file> può contenere un prefisso d'elenco, come in "/test/funz3". Sui sistemi MS-DOS/PC-DOS si può digitare anche una lettera indicante l'unità disco e usare la barra rovesciata "\" (ma bisogna tenere presente che per ottenere una barra rovesciata in una stringa occorre digitare "\\").

Esempi:

```
(setq f (open "/x/nuovo.tst" "w"))   restituisce   <File #nnnn>
(setq f (open "nessun.fil" "r"))     restituisce   NIL
```

4.87 ...)

Ques... :tuisce l'OR logico di una lista di espressioni. OR valuta l
sinis... ricerca di un'espressione con valore diverso da NIL. Quand
espre... :rrompe ogni ulteriore valutazione e restituisce T. Se tut
hannc... R restituisce NIL. Esempi:

```
.5 '())          restituisce   T
 ())             restituisce   NIL
```

4.88 ...t> <stringa modo>)

Que... :ituisce un punto (lista di due numeri reali) risultante dall
moc... getto descritti nella <stringa modo> al punto <pt>. La <strir
strin... uno o più identificatori validi di snap ad oggetto, quali "
ecc... gole. Esempi:

```
2 (osnap pt1 "medio"))
2 (osnap pt1 "medio,fine,centro"))
```

Se l'... è un punto bidimensionale, la funzione restituirà un punt
se ... ento <pt> è un punto tridimensionale, la funzione res
corr... non viene individuato nessun modo di snap ad oggetto ad
spe... unto <pt> dato, viene restituito NIL.

L'e... uesta funzione dipende dalla vista 3D corrente e dall'i...
va... a FLATLAND. Rimandiamo al Capitolo 8 e all'Appendi
all... D per ulteriori informazioni.

4.8...

Qu... a funzione ma è la costante *pi greco*: il suo valore ap
3.1...

4. ... <pt> <angolo> <distanza>)

Q... restituisce il punto dell'UCS con angolo <angolo> e dista
p... UCS. <angolo> è espresso in radianti partendo dall'asse X cc
in... raria. Benché <pt> possa esser tridimensionale, <angolo> è
p... one corrente. Se la variabile di sistema FLATLAND è 0.
p... nti il punto restituito sarà 2D. Ad esempio (se FLATLAND

```
(1.0 1.0 3.5) 0.785398 1.414214)   restituisce
```

4. ... <espr> [<desc file>])

Q... visualizza l'espressione <espr> sullo schermo e restituisce <es
u... ualsiasi e non deve necessariamente essere una stringa di tes
e... un descrittore di file per un file aperto alla scrittura), <espr

da
de
ni
dei
na
o".
ale.
nto
do>
della
uide
e di
> dal
ente
vo al
o un
3.5)
essere
file>
un file

allo stesso modo come apparirebbe sullo schermo. Viene scritta solo l'espressione specificata; non vengono inclusi spazi o nuove righe. Determinando per esempio quanto segue:

```
(setq a 123)
(setq b '(a))
```

ne risulta che:

`(prin1 'a)`	visualizza A	e restituisce	A
`(prin1 a)`	visualizza 123	e restituisce	123
`(prin1 b)`	visualizza (A)	e restituisce	(A)
`(prin1 "Ciao")`	visualizza "Ciao"	e restituisce	"Ciao"

Ognuno di questi esempi visualizza sullo schermo, dato che non è stato specificato un <desc file>. Supponendo che F sia un descrittore di file valido per un file aperto alla scrittura:

```
(prin1 "Ciao" f)
```

scriverebbe "Ciao" nel file specificato e restituirebbe "Ciao".

Se <espr> è una stringa contenente caratteri di controllo, PRIN1 li visualizza preceduti da una barra rovesciata, come segue:

\e	per "escape"
\n	per "newline"
\r	per "return"
\t	per "tab"
\nnn	per il carattere il cui codice ottale è nnn

Quindi:

`(prin1 (chr 2))`	visualizza	"\002"	e restituisce	"\002"
`(prin1 (chr 10))`	visualizza	"\n"	e restituisce	"\n"

PRIN1 può essere utilizzato senza argomenti e restituirà (e visualizzerà) un simbolo il cui nome corrisponde alla stringa nulla. Se utilizzate PRIN1 (senza argomenti) come ultima espressione in una funzione utente, quando la funzione sarà completa otterrete solo la visualizzazione di una linea vuota, ciò che vi permetterà di uscire da una funzione senza procurare incidenti. Facciamo un'esempio:

```
(defun C:IMPOST ()
     (setvar "LUNITS" 4)
     (setvar "PUNTINI" 0)
     (prin1)
)
```

allora:

Comando: <u>IMPOST</u>

questa sequenza eseguirà il comando definito dall'utente, eseguirà le funzioni SETVAR richieste e ripresenterà il messaggio di richiesta comandi senza visualizzare alcun messaggio estraneo.

4.92 (princ <espr> [<desc file>])

Questa funzione è identica a PRIN1, solo che i caratteri di controllo in <espr> sono scritti senza espansioni. In generale, PRIN1 è destinato a scrivere espressioni in modo che siano compatibili con LOAD, mentre PRINC le scrive in modo che siano leggibili da funzioni quali READ-LINE.

4.93 (print <espr> [<desc file>])

Questa funzione è equivalente a PRIN1, solo che prima della <espr> viene introdotta una nuova riga e dopo <espr> viene introdotto uno spazio.

4.94 (progn <espr>...)

Questa funzione valuta ogni <espr> in modo sequenziale e restituisce il valore dell'ultima espressione. Ad esempio:

```
(if (= a b) (progn
                (setq a (+ a 10))
                (setq b (- b 10))
            )
)
```

Normalmente la funzione IF valuta un'espressione "then" se l'espressione "test" dà un valore qualsiasi eccetto NIL. In questo esempio, invece, abbiamo usato PROGN per far valutare due espressioni.

4.95 (prompt <mess>)

Questa funzione visualizza un messaggio sull'area riservata ai messaggi di richiesta dello schermo e restituisce NIL. <mess> è una stringa di testo. Su configurazioni AutoCAD a due schermi. PROMPT visualizza <mess> su entrambi gli schermi e per questa ragione è preferibile a PRINC. Ad esempio:

```
(prompt "Nuovo valore: ")
```

visualizza "Nuovo valore: " sullo schermo e restituisce NIL.

4.96 (quote <espr>)

Questa funzione non valuta l'<espr> e la restituisce letteralmente. Può essere scritta nel modo seguente:

```
'espr
```

Esempi:

`(quote a)`	restituisce	`A`
`(quote cat)`	restituisce	`CAT`
`(quote (a b))`	restituisce	`(A B)`
`'a`	restituisce	`A`
`'cat`	restituisce	`CAT`
`'(a b)`	restituisce	`(A B)`

(Gli ultimi tre esempi non funzionano se sono digitati direttamente dalla tastiera in risposta a un messaggio di AutoCAD. Si tenga presente che un'immissione di questo tipo deve iniziare con "(" o "!" per poter essere riconosciuta da un'espressione LISP.)

4.97 `(read <stringa>)`

Questa funzione restituisce la prima lista (o atomo) ottenuta da <stringa>. <stringa> non deve contenere spazi vuoti. Esempi:

`(read "Ciao")`	restituisce	`CIAO`
`(read "(a)")`	restituisce	`(A)`

4.98 `(read-char [<desc file>])`

Questa funzione legge un singolo carattere dal buffer di input della tastiera o dal file aperto descritto da <desc file>; restituisce un numero intero che corrisponde al codice ASCII rappresentante il carattere letto.

Se non è stato specificato un <desc file> e non ci sono caratteri nel buffer di input della tastiera, READ-CHAR si aspetta che si digiti qualcosa sulla tastiera (seguito da RETURN). Supponendo che il buffer di input della tastiera sia vuoto:

```
(read-char)
```

aspetterà che si immetta qualcosa. Se si digita "ABC" seguito da RETURN, per esempio, READ-CHAR restituisce 65, cioè il codice ASCII che sta per "A". I richiami di READ-CHAR successivi restituiscono rispettivamente 66, 67 e 10 (a capo). Se si richiama ancora una volta READ-CHAR, la funzione aspetta un nuovo input dell'utente.

I sistemi operativi sotto ai quali funzionano AutoCAD e AutoLISP utilizzano convenzioni differenti per segnalare la fine di una linea in un file di testo ASCII. UNIX, ad esempio, utilizza un carattere solo per indicare una nuova linea (LF, codice ASCII 10), mentre PC-DOS/MS-DOS utilizza una coppia di caratteri (CR/LF, codici ASCII 13 e 10). Per facilitare la redazione di programmi AutoLISP trasferibili tra diversi sistemi operativi gestiti da AutoCAD, READ-CHAR accetta tutte le convenzioni e restituisce un carattere solo di nuova linea (codice ASCII 10) ogni volta che incontra un carattere (o una sequenza di caratteri) indicante una fine di linea.

4.99 (read-line [<desc file>])

Questa funzione legge una stringa immessa dalla tastiera o dal file aperto descritto da <desc file>. Se viene incontrata la fine del file, READ-LINE restituisce NIL, in caso contrario restituisce la stringa che è stata letta. Supponendo che F sia un descrittore di file valido:

```
(read-line f)
```

restituirebbe la riga di input successiva del file oppure NIL, se è stata raggiunta la fine del file.

4.100 (redraw [<nome entità> [<modo>]])

L'effetto della funzione dipende dal numero di argomenti fornito. Se viene richiamata senza argomenti:

```
(redraw)
```

ridisegna la finestra corrente come lo farebbe il comando RIDIS (REDRAW) di AutoCAD. Se richiamata con un argomento <nome entità>:

```
(redraw <nome entità>)
```

l'entità selezionata verrà ridisegnata. Ciò è utile per identificare un'entità dopo aver usato GRCLEAR per ripulire lo schermo. I nomi delle entità sono descritti nel Capitolo 5 di questo manuale.

Un controllo totale della funzione che ridisegna un'entità è data richiamando REDRAW con due argomenti:

```
(redraw <nome entità> <modo>)
```

<nome entità> è il nome dell'entità che deve essere ridisegnata e <modo> è un intero con uno dei valori seguenti:

Modo RIDIS	Effetto
1	Ridisegna entità sullo schermo
2	Cancella entità
3	Evidenzia entità
4	Disattiva evidenziazione

Se <nome entità> è l'intestazione di un'entità complessa (polilinea o blocco complesso), l'entità principale e tutte le entità secondarie saranno elaborate se il <modo> è positivo. Se l'argomento <modo> è negativo, solo l'entità d'intestazione verrà elaborata da REDRAW.

REDRAW restituisce sempre NIL.

4.101 (rem <num1> <num2>...)

Questa funzione divide <num1> per <num2> e restituisce il (<num1> mod <num2>) restante. REM può essere usata con numeri interi o reali e valgono le regole standard di conversione. Esempi:

```
(rem 42 12)          restituisce    6
(rem 12.0 16)        restituisce    12.0
(rem 60 3)           restituisce    0
```

4.102 (repeat <numero> <espr>...)

In questa funzione <numero> rappresenta un numero intero e positivo qualsiasi. La funzione valuta l'<espr> un dato <numero> di volte e restituisce il valore dell'ultima espressione. Determinando per esempio quanto segue:

```
(setq a 10)
(setq b 100)
```

allora:

```
(repeat 4
      (setq a (+ a 10)))
      (setq b (+ b 10)))
)                                 restituisce    140
```

4.103 (reverse <lista>)

Questa funzione restituisce <lista> invertendo l'ordine dei suoi elementi. Ad esempio:

```
(reverse '((a) b c))              restituisce    (C B (A))
```

4.104 (rtos <numero> [<modo> [<precisione>]])

Questa funzione restituisce una stringa che è la rappresentazione di <numero> (reale) secondo l'impostazione del <modo>, della <precisione> e della variabile di quotatura DIMZP di AutoCAD. Gli argomenti <modo> e <precisione> sono interi che selezionano la precisione e il modo delle unità lineari. I valori di <modo> sono:

Modo RTOS	Formato
1	Scientifico
2	Decimale
3	Ingegnerile (piedi + pollici decimali)
4	Architetturale (piedi + pollici frazionari)
5	Frazionario

Gli argomenti <modo> e <precisione> corrispondono alle variabili di sistema di AutoCAD LUNITS e LUPREC. Se si omettono gli argomenti, verranno usate le impostazioni correnti di LUNITS e LUPREC.

Ad esempio, se DIMZP = 0:

```
(rtos 17.5 1 4)      restituisce    "1.7500E+01"
(rtos 17.5 2 2)      restituisce    "17.50"
(rtos 17.5 3 2)      restituisce    "1'-5.50""
(rtos 17.5 4 2)      restituisce    "1'-5 1/2""
(rtos 17.5 5 2)      restituisce    "17 1/2"
```

4.105 (set <simb> <espr>)

Questa funzione assegna alla <espr> il valore di <simb> (che è il nome di un simbolo citato) e restituisce quel valore. Esempi:

```
(setq 'a 5.0)          restituisce    5.0     e definisce il simbolo A
(set (quote b) 'a)     restituisce    A       e definisce il simbolo B
```

Se la funzione SET è usata con un nome di simbolo non citato, può assegnare un nuovo valore a un altro simbolo *indirettamente*. Considerando gli esempi precedenti:

```
(set b 640)     restituirebbe   640
```

e assegnerebbe il valore 640 al simbolo A (dato che il simbolo B lo contiene). Rinviamo a SETQ.

4.106 (setq <simb1> <espr1> [<simb2> <espr2>]...)

Questa funzione assegna alla <espr1> il valore di <simb1>, alla <espr2> il valore di <simb2> e così via; è la funzione di assegnamento principale in AutoLISP e resituisce l'ultima <espr>. Per esempio:

```
(setq a 5.0)          restituisce    5.0
```

e dà al simbolo A il valore 5.0. Ogni volta che A viene valutato, restituirà il numero reale 5.0. Altri esempi:

```
(setq b 123 c 4.7)    restituisce    4.7
(setq s "ciò")        restituisce    "ciò"
(setq x '(a b))       restituisce    (A B)
```

Le funzioni SET e SETQ creano o modificano simboli globali, a meno che non vengano usate all'interno di una funzione DEFUN per assegnare un valore all'argomento di una funzione o ad un simbolo definito come locale a quella funzione (DEFUN). Ad esempio:

```
(setq glo1 123)                   (Crea un simbolo globale)
 (defun demo (arg1 arg2 /loc1 loc2)
 (setq arg1 234)                  (attribuisce un nuovo valore locale)
 (set loc1 345)                   (attribuisce un nuovo valore locale)
 (setq glo1 456)                  (attribuisce un nuovo valore globale)
 (setq glo2 567)                  (crea un nuovo simbolo globale)
)
```

Simboli globali possono essere selezionati o modificati da ogni funzione o usati all'interno di ogni espressione. Simboli locali e argomenti di funzioni sono significativi solo durante la valutazione di una funzione che li definisce (e durante funzioni richiamate da questa). Argomenti di funzioni possono essere impiegati in qualità di simboli locali; la funzione può modificare i loro valori, ma tali trasformazioni vengono eliminate appena si esce dalla funzione.

SET e SETQ sono in grado di attribuire nuovi valori a simboli incorporati e nomi di funzioni, annullando le attribuzioni precedenti o rendendole inaccessibili. Molti utenti hanno commesso l'inaccortezza di impostare dati di questo tipo:

```
(setq angolo (...))        ;ERROR!
(setq lunghezza (...))     ;ERROR!
(setq max (...))           ;ERROR!
(setq t(...))              ;ERROR!!
(setq pi 3.0)              ;ERROR!!!
```

Per evitare ogni tipo di errore imprevisto, occorre fare molta attenzione nella scelta dei nomi da dare ai propri simboli. *Non bisogna mai usare per un proprio simbolo lo stesso nome di un simbolo già incorporato in una funzione!* (Per ottenere una lista dei nomi di simboli da evitare, basta digitare "!ATOMLIST" al messaggio "Comando" di AutoCAD prima di caricare qualunque funzione di AutoLISP).

4.107 (setvar <nome var> <valore>)

Questa funzione assegna un <valore> a una variabile di sistema di AutoCAD e restituisce questo valore. Il nome della variabile deve essere scritto fra virgolette. Ad esempio:

```
(setvar "FILLETRAD" 0.50)        restituisce    0.500000
```

cioè specifica un raggio di raccordo in AutoCAD di 0.5 unità. Per le variabili di sistema con valori interi, il <valore> fornito deve essere tra -32768 e +32767.

Alcuni comandi di AutoCAD assumono i valori delle variabili di sistema prima di visualizzare i messaggi di richiesta dati. Se utilizzate MODIVAR per impostare un nuovo valore mentre un comando è attivo, il nuovo valore non sarà effettivo prima del prossimo comando AutoCAD.

Rimandiamo all'Appendice A della Guida all'Uso per la lista delle variabili correnti di AutoCAD. Vedi anche la funzione GETVAR.

4.108 (sin <angolo>)

Questa funzione restituisce il seno di <angolo> sottoforma di numero reale. L'angolo è espresso in radianti. Esempi:

```
(sin 1.0)      restituisce    0.841471
(sin 0.0)      restituisce    0.000000
```

4.109 (sqrt <numero>)

Questa funzione restituisce la radice quadrata del <numero> sottoforma di numero reale. Esempi:

```
(sqrt 4)        restituisce  2.000000
(sqrt 2.0)      restituisce  1.414214
```

4.110 (strcase <stringa> [<espr>])

STRCASE valuta la stringa specificata da <stringa> e restituisce una copia di <stringa> con tutti i caratteri alfabetici convertiti in maiuscole o minuscole a seconda dell'argomento <espr>. Se <espr> non è presente o è NIL, tutti i caratteri alfabetici nella <stringa> saranno convertiti in maiuscole; se invece <espr> è presente e diverso da NIL, tutti i caratteri alfabetici nella <stringa> saranno convertiti in minuscole. Ad esempio:

```
(strcase  "Modello")          restituisce   "MODELLO"
(strcase  "Modello" T)        restituisce   "modello"
```

4.111 (strcat <stringa1> <stringa2>...)

Questa funzione restituisce una stringa che è la concatenazione di <stringa1>, <stringa2>, ecc.. Esempi:

```
(strcat "circo" "scritto")    restituisce   "circoscritto"
(strcat "a" "b" "c")          restituisce   "abc"
(strcat "a" "" "c")           restituisce   "ac"
```

4.112 (strlen <stringa>)

Questa funzione restituisce la lunghezza, in caratteri, di una stringa sottoforma di numero intero. Esempi:

```
(strlen "abcd")     restituisce   4
(strlen "ab")       restituisce   2
(strlen "")         restituisce   0
```

4.113 (subst <nuovo elem> <elem prec> <lista>)

Questa funzione cerca l'elemento <elem prec> nella <lista> e restituisce una copia di <lista> con tutti gli <elem prec> sostituiti da <nuovo elem>. Se <elem prec> non è reperibile nella lista, SUBST restituisce la <lista> invariata. Avendo definito per esempio:

```
(setq modello '(a b (c d) b)
```

allora:

(subst 'qq 'b modello)	restituisce	(A QQ (C D) QQ)
(subst 'qq 'z modello)	restituisce	(A B (C D) B)
(subst 'qq '(c d) modello)	restituisce	(A B QQ B)
(subst '(qq rr) '(c d) modello)	restituisce	(A B (QQ RR) B)
(subst '(qq rr) 'z modello)	restituisce	(A B (C D) B)

Se usato con ASSOC, SUBST permette di sostituire il valore associato a una chiave in una lista di associazioni. Determinando per esempio quanto segue:

```
(setq chi '((nome greta) (cognome garbo) (detta la_divina)))
```

allora:

```
(setq prec
     (assoc 'nome chi)
)                          restituisce    (NOME GRETA)
(setq nuovo '(nome g))     restituisce    (NOME G)
(subst nuovo prec chi)     restituisce    ((NOME G) (COGNOME GARBO)
                                          (DETTA LA_DIVINA))
```

4.114 (substr <stringa> <inizio> [<lung>])

Questa funzione restituisce una sottostringa di <stringa> che inizia nella posizione del carattere <inizio> della <stringa> e continua per la lunghezza <lung> di caratteri. Se <lung> non è specificata, la sottostringa continua fino alla fine della stringa. <inizio> (e <lung>, se presente) deve essere un numero intero positivo. Il primo carattere della <stringa> è il carattere numero 1. Esempi:

(substr "abcde" 2)	restituisce	"bcde"
(substr "abcde" 2 1)	restituisce	"b"
(substr "abcde" 3 2)	restituisce	"cd"

4.115 (terpri)

Questa funzione visualizza un passaggio a nuova linea sullo schermo e restituisce NIL. TERPRI non viene usato per file I/O. Per scrivere un "a capo" in un file, bisogna ricorrere a PRINT o PRINC.

4.116 (textscr)

La funzione TEXTSCR cambia dallo schermo grafico allo schermo testo sui sistemi monoschermo (come il tasto di commutazione "FLIP SCREEN" di AutoCAD). TEXTSCR restituisce sempre NIL. Vedi anche la funzione GRAPHSCR.

4.117 (trace <funzione>)

Questa funzione è un aiuto di messa a punto: segnala le <funzioni> specificate con un identificatore nel percorso del programma e restituisce il nome dell'ultima funzione. Ogni volta che viene valutata la <funzione>, viene visualizzato il percorso del programma con l'immissione della funzione (rientrata al suo livello di profondità) e il risultato della funzione verrà scritto sullo schermo. Ad esempio:

```
(trace mia-funz)          restituisce    MIA-FUNZ
```

e posiziona l'identificatore di percorso per la funzione MIA-FUNZ. Vedi anche UNTRACE.

4.118 (trans <pt> <da> <a> [<spostamento>])

Questa funzione trasferisce un punto (o uno spostamento) da un sistema di coordinate all'altro. L'argomento <pt> è composto da un lista di 3 numeri reali o da un vettore di spostamento a 3 dimensioni. <da> è un codice indicante il sistema di coordinate nel quale <pt> è espresso e <a> è il codice indicante il sistema di coordinate del punto da restituire. L'argomento opzionale <spostamento>, se presente e diverso da 0, specifica che <pt> deve essere considerato come un vettore di spostamento 3D e non come un punto. Gli argomenti <da> e <a> possono essere del tipo seguente:

- Uno dei codici interi della tabella seguente.

Codice	Sistema di coordinate
0	Sistema di Coordinate Globali (WCS)
1	Sistema di Coordinate corrente (UCS)
2	Sistema di Coordinate di visualizzazione della finestra corrente - vedi in basso

- Un nome di entità del tipo restituito dalla funzioni ENTNEXT, ENTLAST, ENTSEL e SSNAME descritte nel Capitolo 5. Questo permette di trasferire un punto da o a un Sistema di Coordinate d'Entità (ECS) di un'entità specifica. (Per alcune entità, l'ECS equivale al WCS, per cui la conversione tra ESC e WCS sarà per queste entità un'operazione nulla).

- Un vettore di estrusione 3D (una lista di 3 numeri reali). Questo è un altro metodo per convertire da o a un ESC. Questo sistema non funziona però se l'ECS è equivalente al WCS.

TRANS restituisce un punto tridimensionale (o uno spostamento) nel sistema di coordinate specificato dall'argomento <a>. Ad esempio, dato un UCS ruotato di 90 gradi in senso antiorario intorno all'asse Z del WCS:

```
(trans '(1.0 2.0 3.0) 0 1)    restituirebbe   (2.0 -1.0 3.0)
(trans '(1.0 2.0 3.0) 1 0)    restituirebbe   (-2.0 1.0 3.0)
```

Segue una descrizione dei sistemi di coordinate gestiti da TRANS e del modo in cui questi vengono gestiti.

WCS Questo è il sistema di coordinate di referenza. Tutti gli altri sistemi di coordinate vengono definiti relativamente a esso. Il WCS è l'unico sistema di coordinate non modificabile. La sua funzione principale è quella di definire coordinate fisse, che non cambiano al passaggio ad un altro sistema di coordinate.

UCS Questo sistema di coordinate viene definito dall'utente per facilitare (o, in certi casi, rendere possibili) alcune operazioni di editazione. Tutte le immissioni di punti (compresi i punti restituiti da un'espressione di AutoLISP ma esclusi i punti preceduti da un "*") sono interpretati relativamente all'UCS corrente. Se volete quindi fornire valori in WCS, ECS o DCS in risposta a comandi di AutoCAD, dovete prima convertirli nei valori corrispondenti nell'UCS corrente tramite TRANS.

ECS Valori di punto restituiti da ENTGET (Capitolo 5) sono misurati relativamente a questo sistema di coordinate. Tali punti sono di poca utilità finchè non vengono convertiti nel sistema di coordinate appropriato all'uso che se ne vuole fare.

Ad esempio, se desiderate disegnare una linea dal punto di inserimento di un testo (senza utilizzare OSNAP) dovreste convertire il punto d'inserimento dell' entità di testo dal suo ECS all'UCS:

```
(trans '(1 2 3) 1 <nome entità di testo) 1)
```

e immettere il risultato in risposta al messaggio "DAl punto:"

DCS Il sistema di coordinate di visualizzazione (Display Coordinate System) è il sistema di coordinate nel quale le immagini vengono trasformate prima di essere visualizzate. La sua origine è il punto di mira (riportato dalla variabile di sistema TARGET) e il suo asse Z è la direzione di vista. Nel DCS ci si trova sempre in vista piana (l'Y punta verso l'alto e l'X è rivolto verso destra). E' utilizzato per determinare come una costruzione appare a chi guarda lo schermo.

Ad esempio, se avete immesso un punto e desiderate individuare a quale estremo di linea il punto *appare* più vicino, vi conviene convertire questo punto dall'UCS al DCS.

```
(trans <punto> 1 2)
```

e ogni estremo di linea dall'ECS della linea al DCS.

```
(trans <estremo> <nome linea> 2)
```

Potete calcolare la distanza tra il punto e ogni estremo della linea (ignorando la componente Z) per determinare quale estremo *sembra* più vicino.

TRANS può trasformare anche punti 2D, *inventando* un valore appropriato per la Z. La componente Z utilizzata dipende dal sistema di coordinate di partenza specificato e se il valore deve essere convertito in un punto o in uno spostamento. La Z viene *inventata* nella maniera seguente:

Da	Punto	Spostamento
UCS	0.0	0.0
UCS	elevazione corrente	0.0
ECS	0.0	0.0
DCS	proiettato sul piano di costruzione corrente (piano XY dell'UCS + elevazione corrente)	0.0

4.119 (type <elemento>)

Questa funzione restituisce il tipo di <elemento>. in cui TYPE è uno dei seguenti:

REAL	numeri a virgola mobile
FILE	descrittori di file
STR	stringa di caratteri
INT	numeri interi
SYM	simboli
LIST	liste (e funzioni utente)
SUBR	funzioni interne di AutoLISP
PICKSET	gruppi di selezione AutoCAD
ENAME	nomi di entità AutoCAD
PAGETB	tabella di paginazione di funzioni

Per esempio, determinando quanto segue:

```
(setq a 123 r 3.45 s "Ciao" x '(a b c))
(setq f (open "nome" "r"))
```

ne risulta che:

```
(type 'a)        restituisce    SYM
(type a)         restituisce    INT
(type f)         restituisce    FILE
(type r)         restituisce    REAL
(type s)         restituisce    STR
(type x)         restituisce    LIST
(type +)         restituisce    SUBR
```

L'esempio seguente illustra un'applicazione della funzione TYPE.

```
(defun intero (a)
    (se (= (type a) 'INT)       ;type è un intero?
        T                       ;si, restituisce T
        NIL                     ;no, restituisce NIL
    )
)
```

4.120 (untrace <funzione>...)

Questa funzione rimuove l'identificatore di percorso per le <funzioni> specificate e restituisce il nome dell'ultima funzione. Essa disattiva selettivamente l'aiuto di messa a punto del percorso del programma. Per esempio:

`(untrace mia-funz)` restituisce `MIA-FUNZ`

e cancella l'identificatore di percorso per MIA-FUNZ. Vedi anche TRACE.

4.121 (ver)

Questa funzione restituisce una stringa che contiene il numero corrente della versione di AutoLISP e dovrebbe essere usata assieme a EQUAL per verificare la compatibilità dei programmi. La stringa si presenta di questo tipo:

```
"AutoLISP versione X.X"
```

in cui X.X è il numero della versione. Esempio:

`(ver)` potrebbe restituire `"AutoLISP versione 10.0"`

In AutoLISP Esteso, VER restituisce una stringa della forma seguente:

```
"AutoLISP Esteso Release X.X"
```

se lavorate con un'applicazione, potrete quindi sapere se vi state servendo dell'AutoLISP normale o di quello esteso consultando la stringa restituita da VER.

4.122 (vports)

Questa funzione restituisce una lista dei descrittori delle finestre attive al momento. Un descrittore di finestre è una lista consistente del numero di identificazione della finestra e delle posizioni degli angoli in basso a sinistra e in alto a destra della finestra. Questi angoli sono espressi in valori da 0.0 a 1.0 con (0.0 0.0) rappresentante l'angolo in basso a sinistra dell'area grafica del monitor e (1.0 1.0) l'angolo in alto a destra.

Ad esempio, data una configurazione a finestra unica, la funzione VPORTS restituirà:

```
((1 (0.0 0.0) (1.0 1.0)
```

Parimenti, date quattro finestre di dimensioni uguali collocate ai quattro angoli dello schermo. VPORTS restituirà:

```
( (5 (0.5 0.0) (1.0 0.5)
  (2 (0.5 0.5) (1.0 1.0)
  (3 (0.0 0.5) (0.5 1.0)
  (4 (0.0 0.0) (0.5 0.5)) )
```

Il descrittore della finestra corrente è sempre il primo della lista. Nell'esempio precedente, la finestra numero 5 è la finestra corrente.

4.123 (while <espr test> <espr>...)

WHILE valuta <espr test> e se questa è diversa da NIL, valuta le altre espressioni e rivaluta ancora una volta <espr test>. Questo procedimento continua finché <espr test> è NIL. A questo punto WHILE restituisce il valore più recente dell'ultima espressione. Dato ad esempio:

```
(setq a 1)
```

ne risulta che:

```
(while (<= 10)
       (funz-utente a)
       (setq a (1+ a)
)
```

richiamerebbe dieci volte la funzione utente FUNZ-UTENTE assegnando ad A i valori da 1 a 10. In seguito restituirebbe 11, che è il valore dell'ultima espressione valutata.

4.124 (write-char <num> [<desc file>])

Questa funzione scrive un carattere sullo schermo o in un file aperto descritto da <desc file>. <num> è il codice ASCII che rappresenta il carattere ed è anche il valore restituito dalla funzione. Ad esempio:

`(write-char 67)` restituisce 67

e scrive la lettera C sullo schermo. Supponenedo che F sia il descrittore di un file aperto, allora:

`(write-char 67 f)` restituisce 67

e scrive la lettera C in quel file.

I sistemi operativi sotto ai quali funzionano AutoCAD e AutoLISP utilizzano convenzioni differenti per segnalare la fine di una linea in un file di testo ASCII. UNIX, ad esempio, utilizza un solo carattere per indicare una nuova linea (LF, codice ASCII 10), mentre PC-DOS/MS-DOS utilizza una coppia di caratteri (CR/LF, codici ASCII 13 e 10). Per facilitare la redazione di programmi AutoLISP trasferibili tra diversi sistemi operativi gestiti da AutoCAD, WRITE-CHAR accetta tutte le convenzioni e restituisce un solo carattere di nuova linea (codice ASCII 10) ogni volta che incontra un carattere (o una sequenza di caratteri) indicante una fine di linea. Ad esempio, su sistemi PC-DOS/MS-DOS:

`(write-char 10 f)` restituisce 10

ma scrive nel file la sequenza di caratteri CR/LF (codici ASCII 13 e 10). WRITE-CHAR non può scrivere un carattere nullo (codice ASCII 0) in un file.

4.125 (write-line <stringa> [<desc file>])

Questa funzione scrive una stringa sullo schermo o in un file aperto descritto da <desc file>. Essa restituisce la stringa citata come di consueto ma omette le virgolette quando la scrive in un file. Supponendo che F sia un descrittore valido di un file aperto, allora:

```
(write-line "Test" f)      scrive Test      e restituisce   "Test"
```

4.126 (zerop <elemento>)

Questa funzione restituisce T se <elemento> è un numero reale o intero che dà zero, in caso contrario restituisce NIL. Non è definita per altri tipi di elementi. Esempi:

```
(zerop 0)          restituisce   T
(zerop 0.0)        restituisce   T
(zerop 0.0001)     restituisce   NIL
```

4.127 (*error* <stringa>)

Questa è una funzione per la gestione di errori ed è definibile dall'utente; se non è NIL, viene eseguita come funzione ogni volta che si avvera una condizione di errore in AutoLISP. Il suo unico argomento è una stringa che contiene la descrizione dell'errore. Esempio:

```
(defun *error* (st)
     (princ "errore: ")
     (princ st)
     (terpri)
)
```

Questa funzione ha esattamente lo stesso effetto del gestore di errori standard di AutoLISP: visualizza "errore:" e la sua descrizione.

Capitolo 5

Accesso a entità e dispositivi

Tramite un gruppo di funzioni di AutoLISP è possibile accedere alle entità di AutoCAD, allo schermo grafico e ai dispositivi di immissione dati. Si possono selezionare entità, esaminarne i valori e modificarli. Inoltre è possibile memorizzare gruppi di selezione in variabili LISP per operare con insiemi di entità. Dato che la funzione COMMAND permette di sottoporre comandi ad AutoCAD per creare direttamente delle entità, non sono previste funzioni speciali per questo scopo.

5.1 Tipi di dati speciali

Per rendere possibile l'accesso a entità AutoCAD, sono stati implementati in AutoLISP due tipi di dati speciali: *nome d'entità* e *gruppo di selezione*. Questi tipi di dati vengono manipolati solo dalle funzioni che li elaborano e la loro struttura interna non è rilevante per la programmazione in LISP. Il nome d'entità è un indicatore in un file mantenuto dall'Editore di Disegni, dal quale AutoLISP è in grado di reperire il registro della base di dati grafica dell'entità e i suoi vettori (se sono sul monitor). Un gruppo di selezione è semplicemente un insieme di nomi d'entità.

I nomi di entità e i gruppi di selezione sono validi solo per la sessione dell'editore di disegni durante la quale vengono ottenuti da AutoCAD.

Inoltre, funzioni relative ad entità possono richiamare e manipolare gli *identificatori di entità* permanenti che AutoCAD associa ad ogni entità. AutoLISP interpreta questi identificatori come stringhe composte da cifre esagesimali.

5.2 Funzioni relative al gruppo di selezione

Le funzioni che seguono effettuano varie operazioni con il gruppo di selezione.

5.2.1 `(ssget [<modo>] [<pt1> [<pt2>]])`

La funzione SSGET restituisce un gruppo di selezione. L'argomento opzionale <modo> è una stringa che specifica il tipo di selezione entità. Può trattarsi di "F", "I", "U" o "P" che corrispondono ai modi di selezione di AutoCAD "Finestra", "Interseca", "Ultimo" o "Precedente". Gli argomenti <pt1> e <pt2> specificano i punti rilevanti per la selezione sottoforma di liste. Specificare un punto senza l'argomento <modo> equivale a selezionare un'entità mediante puntamento. Se mancano tutti gli argomenti, SSGET presenta il dialogo generale della funzione "Selezionare oggetti:" di AutoCAD permettendo una costruzione interattiva del gruppo di selezione. I gruppi di selezione SSGET contengono solo entità vere e proprie (nessun attributo e vertice di polilinea). Esempi:

`(ssget)`	*sollecita l'utente ad eseguire una selezione*
`(ssget "P")`	*seleziona il gruppo di selezione precedente*
`(ssget "U")`	*seleziona l'ultima entità aggiunta al database*
`(ssget '(2 2))`	*seleziona l'entità passante per il punto 2.2*
`(ssget "F" '(0 0) '(5 5))`	*seleziona le entità nella finestra da 0.0 a 5.5*
`(ssget "I" '(0 0) '(1 1))`	*seleziona le entità intersecanti la finestra*

da 0.0 a 1.1

```
(ssget "X" <lista per filtro>)
```

seleziona le entità idonee alla <lista per filtro>

Gli oggetti selezionati sono evidenziati solo quando SSGET viene usata senza argomenti. Non vengono registrate informazioni sul modo in cui l'entità era stata selezionata (rimandiamo a ENTSEL che prevede questa possibilità). I gruppi di selezione consumano spazio nei file temporanei, quindi in AutoLISP non possono essere aperti più di sei gruppi per volta. Quando si raggiunge questo limite, AutoCAD non crea più gruppi di selezione e restituisce NIL a tutti i richiami di SSGET. Per chiudere un gruppo di selezione di cui non si ha più bisogno, basta assegnargli valore NIL. (Se si è raggiunto il limite di 6 gruppi di selezione aperti, bisogna richiamare la funzione GC affinche un'altra SSGET possa operare).

E' possibile trasmettere ad AutoCAD una variabile di gruppo di selezione in risposta a un messaggio qualsiasi di selezione oggetti per cui sia valida la selezione "Ultimo". Questa variabile selezionerà tutte le entità nel gruppo di selezione.

Filtri SSGET (ssget "X")

Il modo "X" della funzione SSGET esamina l'intero disegno e produce un gruppo di selezione contenente i nomi di tutte le entità che rispondo a criteri specificati. Ad esempio, utilizzando questo dispositivo, è possibile ottenere un gruppo di selezione comprendente tutte le entità di un tipo dato, su un piano dato o di un colore dato.

Questo nuovo modo SSGET si imposta nella maniera seguente:

```
(ssget "X" <lista per filtro>)
```

dove <lista per filtro> indica una lista di coppie puntate, paragonabile al tipo di lista restituito dalla funzione ENTGET. La <lista per filtro> specifica quali delle proprietà delle entità devono essere ricercate e quali sono i valori validi per entrare nel gruppo di selezione. Ad esempio:

```
(ssget "X" '((0 . "CERCHIO")))
```

restituirà un gruppo di selezione che comprenderà tutti i cerchi presenti nel disegno (il codice di gruppo 0 è il tipo dell'entità). Parimenti:

```
(ssget "X" '((8 . "PIANO3")))
```

restituirà un gruppo di selezione che comprenderà tutte le entità contenute nel piano "PIANO3" (il codice di gruppo 8 è il nome del piano). Se viene specificata più di una coppia puntata nella <lista per filtro>, un'entità, per essere inclusa nel gruppo di selezione, deve soddisfare tutte le condizioni specificate. Ad esempio:

```
(ssget "X" '((0 . "CERCHIO") (8 . "PIANO1") (62 . 1)))
```

restituirà un gruppo di selezione contenente tutti i cerchi di colore rosso compresi nel piano "PIANO1".

Benchè la lista per filtro delle entità abbia lo stesso aspetto di una lista restituita da ENTGET, solo una parte dei codici per i gruppi di entità viene riconosciuto da SSGET "X" (vedi la tavola seguente). La funzione SSGET "X" agisce esaminando l'intero disegno e paragonando i campi di ogni entità con i requisiti contenuti nelle lista per filtro. Se le proprietà di un'entità

corrispondono a tutti i requisiti contenuti in una lista per filtro, quest'entità viene inclusa nel gruppo di selezione che verrà restituito alla fine dell'indagine. SSGET restituisce NIL se non viene trovata nella base di dati nessuna entità che corrisponda a tutti i criteri richiesti. Una lista per filtro vuota o incompleta fa scegliere a SSGET tutte le entità contenute nella base di dati.

I seguenti codici per i gruppi d'entità vengono accettati da SSGET "X":

Codice	Significato
0	Tipo dell'entità
2	Nome del blocco per referenza (INSER)
6	Nome del tipo di linea
7	Nome dello stile di testo per Testo, Attributi e Definizione degli Attributi
8	Nome del piano
38	Elevazione
39	Spessore
62	Numero del colore
66	Segnalatore degli attributi
210	Vettore della direzione di estrusione 3D (lista di 3 numeri reali)

Per controllare i codici di gruppo annotati come (numero reale), dovete fornire un numero reale per il valore di controllo. Quindi, per controllare le entità aventi un'elevazione di 2.0:

```
(ssget "X" '((38 . 2)))        ; non sarebbe corretto
(ssget "X" '((38 .2.0)))       ; sarebbe corretto
```

SSGET "X" restituirà NIL se la lista per filtro comprende codici di gruppi d'entità che non sono compresi in questa tavola. Questo garantisce che i programmi di AutoLISP che utilizzano SSGET "X" continueranno a funzionare anche quando verranno in futuro aggiunti nuovi codici.

5.2.2 (sslength <gs>)

Questa funzione restituisce un numero intero che rappresenta il numero di entità presenti nel gruppo di selezione <gs>. Il numero viene restituito in forma reale se è maggiore di 32767. I gruppi di selezione non contengono mai selezioni doppie di un'entità.
Ad esempio:

```
(setq sset (ssget "L"))    ; pone l'ultimo oggetto nel gruppo di selezione
                           SSET
(sslenght sset)            ; restituisce 1
```

5.2.3 (ssname <gs> <indice>)

Questa funzione restituisce il nome d'entità dell'<indice>simo elemento del gruppo di selezione. Se <indice> è negativo o maggiore dell'entità numerata più alta nel gruppo di selezione, viene restituito NIL. Il primo elemento nel gruppo ha l'indice 0. I nomi d'entità nel gruppo di

selezione ottenuti con SSGET saranno sempre nomi di entità principali; le entità secondarie (attributi di blocchi e vertici di polilinee) non verranno restituite (vedi ENTNEXT più avanti che permette di accedere alle entità secondarie).

Ad esempio:

```
(setq sset (ssget))           ; crea un gruppo di selezione chiamato SSET
(setq ent1 (ssname sset 0))   ; restituisce il nome della prima entità in SSET
(setq ent4 (ssname sset 3))   ; restituisce il nome della quarta entità in SSET
```

Per accedere ad entità al di là della 32767esima nel gruppo di selezione, dovete fornire l'argomento dell'indice in forma reale. Ad esempio:

```
(setq entx (ssname sset 50843.0))   ; restituisce il nome della 50844esima
                                      entità di SSET
```

5.2.4 (ssadd [<nome ent> [<gs>]])

Se la funzione è richiamata senza argomenti, SSADD costruisce un nuovo gruppo di selezione senza entità. Se richiamata con un argomento contenente un solo nome di entità, SSADD costruisce un nuovo gruppo di selezione contenente quel singolo nome d'entità; se richiamata con un nome di entità e un gruppo di selezione, essa aggiunge l'entità denominata al gruppo. SSADD restituisce sempre il nuovo gruppo di selezione o quello modificato. Si noti che quando viene aggiunta un'entità a un gruppo, la nuova entità viene aggiunta fisicamente al gruppo esistente e che il gruppo dato come argomento <gs> viene restituito come risultato. Ne consegue che se il gruppo è assegnato ad altre variabili, queste rifletteranno l'aggiunta. Se l'entità denominata si trova già nel gruppo di selezione, l'operazione di SSADD sarà ignorata; non verrà riportato nessun errore. Facciamo alcuni esempi:

```
(setq e1 (entnext))      ; assegna alla prima entità nel disegno il nome
                           E1
(setq ss (ssadd))        ; assegna ad un gruppo di selezione nullo il
                           nome SS
(ssadd e1 ss)            ; restituisce il gruppo di selezione SS con
                           aggiunta l'entità E1
(setq e2 (entnext e1))   ; restituisce l'entità successiva ad E1
(ssadd e2 ss)            ; restituisce il gruppo di selezione SS con
                           aggiunta l'entità E2
```

5.2.5 (ssdel <nome ent> <gs>)

SSDEL cancella il nome d'entità <nome ent> dal gruppo di selezione <gs> e restituisce il nome del gruppo di selezione <gs>. Occorre notare che l'entità viene cancellata fisicamente dal gruppo di selezione e che ciò non corrisponde alla restituzione di un nuovo gruppo di selezione con l'entità cancellata. Se l'entità non si trova nel gruppo di selezione, viene restituito NIL.

Ad esempio, poniamo che l'entità E1 sia un membro del gruppo di selezione SS mentre l'entità E2 non lo è, allora:

```
(ssdel e1 ss)            ; restituisce il gruppo di selezione SS senza
                           l'entità E1
(ssdel e2 ss)            ; restituisce NIL (non modifica SS)
```

5.2.6 (ssmemb <nome ent> <gs>)

Questa funzione verifica se il nome d'entita <nome ent> è un membro del gruppo di selezione <gs>; se è il caso, SSMEMB restituisce il nome d'entita <nome ent>, in caso contrario, restituisce NIL.
Ad esempio, poniamo che l'entità E1 sia un membro del gruppo di selezione SS mentre l'entità E2 non lo è, allora:

```
(ssmemb e1 ss)          ; restituisce il nome dell'entità E1
(ssmemb e2 ss)          ; restituisce NIL
```

5.3 Funzioni relative ai nomi d'entità

Le funzioni che seguono eseguono varie operazioni relative ai nomi d'entità. I nomi d'entita possono essere trasmessi ad AutoCAD in risposta a un qualsiasi messaggio "Selezionare oggetti:" che accetta "Ultimo" come risposta valida. Il risultato sara identico ad una selezione dell'entità tramite finestra.

5.3.1 (entnext [<nome ent>])

Se richiamata senza argomenti, questa funzione restituisce il nome della prima entità non eliminata dalla base dati. Se ENTNEXT è richiamata con l'argomento <nome ent>, restituisce il nome della prima entità non cancellata che *segue* <nome ent> nella base dati. Se non c'è un'entità *successiva* nel data base, viene restituito NIL. ENTNEXT restituisce sia entità principali che secondarie.

Le entità selezionate da SSGET sono entità principali, quindi non sono nè attributi di blocchi nè vertici di polilinee. Si può accedere alla struttura interna di queste entità complesse percorrendo semplicemente le entità secondarie con ENTNEXT. Avendo ottenuto il nome di un'entita secondaria, si può operare con essa come con ogni altra entità. Dopo aver ottenuto il nome di un'entità secondaria mediante ENTNEXT, si puo reperire l'entità principale corrispondente percorrendo le entità con ENTNEXT finchè viene trovata un'entità SEQEND, estraendo poi il gruppo -2 da quell'entità, ciò che equivale al nome dell'entità principale.
Ad esempio:

```
(setq e1 (entnext))        ; assegna alla prima entità del disegno il nome
                           E1
(setq e2 (entnext e1))     ; assegna all'entità successiva ad E1 il nome E2
```

5.3.2 (entlast)

Questa funzione restituisce il nome dell'ultima entità principale non eliminata dal data base. Questa funzione è usata frequentemente per ottenere il nome di una nuova entità che è stata appena aggiunta tramite la funzione COMMAND. Per essere selezionata, l'entità non deve necessariamente trovarsi sullo schermo nè su un piano scongelato.

Ad esempio:

```
(setq e1 (entlast))          ; assegna all'ultima entita del disegno il nome
                             ; E1
(setq e2 (entnext e1)        ; attribuisce a E2 valore NIL (o assegna il
                             ; nome di un attributo o subentità di vertice)
```

Se avete bisogno del nome dell'ultima entità non cancellata (entità vera e propria o subentità), vi conviene definire una funzione del tipo seguente piuttosto che richiamare ENTLAST.

```
(defun lastent (/a b)
   (if (setq a (entlast)                  ; trova l'ultima entita vera e propria
       (while (setq b (entnext a))        ; se seguono subentita
          (setq a b)                      ; ripete finchè non trova piu nessuna
                                          ; entità
       )
   )
   a                                      ; restituisce l'ultima entita o
                                          ; subentita
)
```

5.3.3 (entsel [<messaggio>])

A volte è preferibile selezionare un'entità e, contemporaneamente, specificare il punto con il quale è stata selezionata, come ad esempio nei comandi di AutoCAD OSNAP, SPEZZA, ESTENDE e TAGLIA. La funzione ENTSEL permette ai programmi di AutoLISP di eseguire questa operazione. ENTSEL seleziona un'entità singola richiedendo una selezione mediante puntamento e restituisce una lista il cui primo elemento è il nome dell'entita selezionata e il secondo è la coppia di coordinate del punto di selezione (relative all'UCS corrente). Se per <messaggio> è stata specificata una stringa, essa verrà usata per sollecitare l'utente a immettere l'entità. Se invece non è specificata, il messaggio standard sara "Selezionare oggetti:". La sequenza che segue mostra l'uso della funzione ENTSEL in AutoCAD e la lista che restituisce.

Comando: <u>LINEA</u>
Dai punto: <u>1,1</u>
Al punto: <u>6,6</u>
Al punto: <u>RETURN</u>

Comando: <u>(setq e (entsel "Selezionare un'entità: "))</u>
Selezionare un'entita: <u>3,3</u>
(<Nome entità: 60000014> <u>(3.0 3.0 0.0)</u>

Una lista come questa restituita da ENTSEL può essere fornita ad AutoCAD in risposta a uno qualsiasi dei suoi messaggi di selezione oggetti e verrà interpretata da AutoCAD come punto di selezione dell'entità specificata.

5.3.4 (handent <identificatore>)

Il nome di un'entità può cambiare da una sessione di editazione all'altra, un identificatore di entità rimane però costante per tutto il periodo di esistenza di un'entità. L'argomento <identificatore> è una stringa contenente l'identificatore d'entità, la funzione HANDENT restituisce il nome dell'entità associata con quell'identificatore nella sessione di editazione

```
(handent "5A2")        restituirebbe <nome dell'entità: 60004722>
```

in una particolare sessione di editazione. La stessa chiamata può restituire in un'altra sessione di editazione del medesimo disegno, un nome di entità differente. Può trattarsi della medesima entità e il suo identificatore avrà di conseguenza sempre lo stesso nome ma il nome dell'entità può modificarsi di sessione in sessione.

5.4 Funzioni relative ai dati di entità

Le funzioni che seguono permettono di richiamare e modificare i dati che definiscono un'entità e utilizzano tutte i nomi d'entità per specificare le entità su cui operano.

5.4.1 (entdel <nome ent>)

L'entità specificata dal <nome ent> viene cancellata se si trova nel disegno e ripristinata se è stata cancellata nella stessa sessione. Entità cancellate sono eliminate dal disegno quando si esce dall'Editore di Disegni, quindi ENTDEL puo ripristinarle solo nella sessione in cui erano state cancellate. ENTDEL opera unicamente su entità principali; gli attributi e i vertici di polilinee non possono essere cancellati indipendentemente dalle entità a cui si riferiscono. Per ottenere questo tipo di cancellazione bisogna ricorrere alla funzione COMMAND e richiamare i comandi EDITATT o EDITPL.
Ad esempio:

```
(setq e1 (entnext))     ; assegna alla prima entità del disegno il nome
                          E1
(entdel e1)             ; cancella l'entità E1
(entdel e1)             ; ristabilisce l'entità cancellata E1
```

5.4.2 (entget <nome ent>)

L'entità il cui nome è <nome ent> viene richiamata dalla base dati e restituita sottoforma di lista contenente i dati che la definiscono. La lista risultante ha la forma di una lista di associazioni, dalla quale è possibile estrarre elementi con la funzione ASSOC. Alle entità contenute nella lista risultante sono assegnati i codici di gruppo del file DXF di AutoCAD per ogni parte dei dati che definiscono l'entità.

Per l'esempio seguente partiamo dai seguenti presupposti:

- il piano corrente è "0"
- il tipo di linea corrente è "CONTINUO" (valore standard)
- l'elevazione corrente è 0 (valore standard)
- gli identificatori di entità sono disattivati e
- la variabile di sistema FLATLAND è 0

Disegniamo una linea con questa sequenza:

Comando: <u>LINEA</u>
Dal punto: <u>1,2</u>
Al punto: <u>6,6</u>
Al punto: <u>RETURN</u>

I dati d'entità relativi alla linea possono essere richiamati immettendo:

Comando: <u>(setq a (entget (entlast)))</u>

Presupponendo che FLATLAND è 0 (vengono richiesti punti 3D), si darà ad A il valore della lista seguente:

```
(  (-1 . <Nome entità: 60000014>)
   (0 . "LINEA")                      ; Tipo d'entità
   (8 . "0")                          ; Piano
   (10 1.0 2.0 0.0)                   ; Punto iniziale
   (11 6.0 6.0 0.0)                   ; Punto finale
)
```

L'elemento -1 all'inizio della lista contiene il nome d'entità rappresentato da questa lista. La funzione ENTMOD descritta più avanti, usa questo elemento per identificare l'entità che deve essere modificata.

Le singole coppie puntate che rappresentano i valori possono essere richiamate facilmente con ASSOC e CDR viene usato per estrarne i valori. I codici che compongono l'entità sono quelli usati da DXF e sono riportati nell'Appendice C della Guida all'Uso di AutoCAD. Come per i file DXF, gli elementi d'intestazione delle entità (colore, tipolinea, spessore, flag d'entità complessa, identificatore di entità) sono estratti solo se sono diversi dai valori standard. I campi opzionali di definizione di entità, invece, vengono estratti in entrambi i casi: sia quando corrispondono ai loro valori standard che quando non corrispondono. L'intento è quello di semplificare i programmi di elaborazione, dato che i loro algoritmi generali possono sempre supporre che questi campi siano presenti. Quindi, a differenza di DXF, le coordinate X, Y, Z associate appaiono raggruppate in una lista come (10 1.0 2.0 3.0) e non come gruppi 10, 20 e 30 separati.

Si noti che le sottoliste che specificano punti non sono coppie puntate. Per convenzione, il CDR della sottolista rappresenta il valore del gruppo. Dato che un punto è una lista di due (o tre) numeri interi, il gruppo intero diventa una lista di tre (o quattro) elementi. Il CDR del gruppo è la lista rappresentante il punto, quindi si tiene conto della convenzione secondo cui CDR restituisce sempre il valore.

Quando si scrivono funzioni che elaborano queste liste di entità, bisogna renderle insensibili all'ordine delle sottoliste, usando ASSOC. Il gruppo -1 contenente il nome d'entità permette alle operazioni di modifica di accettare semplicemente la lista d'entità, evitando così la necessità di una struttura parallela che gestisca il nome d'entità. Un'entità SEQEND al termine di una polilinea o di un insieme di attributi contiene un gruppo -2 il cui CDR è il nome d'entità dell'intestazione di questa entità. In questo modo l'intestazione può essere reperita in un'entità secondaria avanzando fino a SEQEND e utilizzando in seguito il CDR del gruppo -2 per trovare il nome d'entità dell'entità principale.

L'esempio che segue illustra la rappresentazione di un'entità complessa sottoforma di lista. Per questo esempio siamo partiti dal presupposto che l'UCS corrente è ruotato di 40 gradi in senso antiorario intorno all'asse X del WCS, che gli identificatori di entità sono attivati e che la variabile di sistema FLATLAND è 0.

Comando: TLINEA
?/Creare/Richiamare/Selezionare: SELEZIONARE
Nuovo tipolinea per entità <DAPIANO>: TRATTEGGIATA
?/Creare/Caricare/Impostare: RETURN

Comando: COLORE
Nuovo colore per entità <DAPIANO>: BLU

Comando: PIANO
?/Def/Piano corrente/Nuovo/ON/OFF/Colore/Tipolinea/conGelare/Scongelare: DEF
Nuovo piano corrente <0>: SCHEMA
?/Def/Piano corrente/Nuovo/ON/OFF/Colore/Tipolinea/conGelare/Scongelare: RETURN

Comando: TESTO
Punto iniziale o Lunghezza/Centrato/Fisso/Mezzo/Destra/Stile: 2,2
Altezza <0.2000>: .3
Angolo di rotazione <0>: 30
Testo: Schema idraulico

Comando: (setq e (entget (entlast)))

In questo caso assegnamo ad E il nome dell'entita di testo e ED corrispondera alla lista che segue. Rimandiamo all'Appendice C della *Guida all'Uso di AutoCAD* per chiarire il senso della lista seguente.

```
(  (-1 . <Nome entità:  6000003C>)
   (0 . "TESTO")                          ; Tipo di entità
   (8 . "SCHEMA")                         ; Piano
   (6 . "TRATTEGGIATA")                   ; Tipolinea
   (62 . 5)                               ; Colore
   (5 . "7E")                             ; Identificatore
   (10 2.0 2.0)                           ; Punto iniziale
   (40 . 0.3)                             ; Altezza
   (1 . "Schema idraulico")
   (50 . 0.523599)                        ; Angolo di rotazione (radianti)
   (41 . 1.0)                             ; Fattore di spessore
   (51 . 0.0)                             ; Angolo obliquo
   (7. "STANDARD")                        ; Stile di testo
   (71 . 0)                               ; Flags di generazione
   (72 . 0)                               ; Giustificazione
   (11 0.0 0.0)                           ; Punto di allineamento
   (210 0.0 -0.642788 0.766044)           ; Vettore direzione di estrusione
)
```

Tutti i punti associati con un'entità sono espressi relativamente al Sistema di Coordinate d'Entità (ECS). Per punti, linee, linee 3D, facce 3D, polilinee 3D, reti 3D e entità di quotatura, l'ECS equivale al WCS (i punti dell'entità sono in coordinate globali). Per tutte le altre entità, l'ECS può essere derivata dal WCS e dalla direzione di estrusione dell'entità (il suo gruppo 210). Quando si opera con entità che sono state disegnate con sistemi di coordinate differenti dal WCS (come il testo dell'esempio precedente), si rivelerà necessario convertire i punti in coordinate globali o coordinate dell'UCS corrente utilizzando la funzione TRANS. Utilizzando l'entità testo di cui sopra come esempio, avremo:

(setq p (cdr (assoc 10 ed))) restituirebbe (2.0 2.0 3.5)

e definirebbe P come il prossimo punto iniziale relativamente all'ECS dell'entità di testo. (Notate che il punto viene restituito in questa forma indipendentemente dall'UCS corrente al momento dell'ENTGET).

(trans p e 0) restituirebbe (2.0 1.53209 1.28558)

E (il nome dell'entità di testo) viene utilizzato come codice di conversione di partenza e trasforma le coordinate ECS del punto iniziale del testo in coordinate globali.

Se la variabile di sistema FLATLAND è diversa da 0 e l'entità in questione è stata disegnata in un UCS equivalente al WCS, ENTGET restituisce i suoi punti come punti bidimensionali del WCS con un gruppo 38 che definisce l'elevazione Z dell'entita (se diversa da 0). In questi casi il gruppo 210 (direzione di estrusione) non viene restituito.

5.4.3 (entmod <lista ent>)

ENTMOD riceve una lista (<lista ent>) nel formato restituito da ENTGET e aggiorna la base dati dell'entità specificata dal gruppo -1 in <lista ent>. Quindi il metodo principale con cui LISP aggiorna la base dati è: richiamare l'entità con ENTGET, modificare la lista che definisce l'entità (si noti che la funzione SUBST è molto utile a questo scopo) e aggiornare infine l'entità nel data base tramite ENTMOD.

Ad esempio:

```
(setq en (entnext))                 ; assegna alla prima entità del disegno il nome
                                      EN
(setq ed (entget en))               ; assegna al dato per il nome dell'entità EN il
                                      nome ED
(setq ed
      (subst (cons 8 "0")
             (assoc 8 ed)           ; modifica il gruppo relativo al piano in ED
           ed                       ; e lo fa corrispondere al piano "0"
      )
)
(entmod ed)                         ; modifica il piano dell'entità EN nel disegno
```

ENTMOD impone alcune restrizioni concernenti i parametri da modificare. In primo luogo, il tipo di un entità e l'identificatore non possono essere modificati. (Se si vuole cambiarlo, bisogna cancellare l'entità con ENTDEL e crearne una nuova con COMMAND). AutoCAD deve poter riconoscere tutti gli elementi ai quali fa riferimento la lista d'entità prima che ENTMOD venga eseguita. Quindi lo stile di testo, il tipo di linea, i nomi di forme o di blocchi devono già essere stati definiti in un disegno prima che possano essere usati in una lista d'entità con ENTMOD. Fanno eccezzione i nomi dei piani. Infatti ENTMOD creerà un nuovo piano con i valori standard dati dal comando "PIANO NUOVO" ogni volta che incontra un riferimento a un piano non definito in una lista d'entità.

Quando modificate i campi d'entità contenenti valori a virgola mobile, (come ad esempio spessore), ENTMOD accetta un numero reale per il nuovo valore e lo converte in valore a virgola mobile. Se fornite un numero a virgola mobile per un campo di entità intero (come il numero di colore), ENTMOD lo tronca e lo converte in un intero.

ENTMOD esegue lo stesso controllo di consistenza sulla lista fornita, come quello di DXFIN sui dati che provengono dal file DXF. Se viene individuato un errore grave che impedisce di aggiornare il data base, viene restituito NIL. In caso contrario, ENTMOD restituisce la lista data come argomento. ENTMOD non modifica campi interni quali il nome d'entità nel gruppo -2 di un'entità SEQEND; tentativi di modificare tali campi non sono presi in considerazione.

Quando viene aggiornata un'entità principale, ENTMOD modifica l'entità e aggiorna l'immagine sullo schermo (entità secondarie comprese). Quando si usa ENTMOD per aggiornare un'entità secondaria, questa verrà aggiornata nel data base ma l'immagine sullo schermo rimarrà invariata. Dopo aver apportato tutte le modifiche alle entità secondarie, si può ricorrere alla funzione ENTUPD descritta qui di seguito per aggiornarne anche l'immagine sullo schermo.

5.4.4 (entupd <nome ent>)

Quando si modifica il vertice di una polilinea o l'attributo di un blocco con ENTMOD, l'entità principale non viene aggiornata sullo schermo. Ad esempio, se per modificare 100 vertici di una polilinea complessa, la polilinea dovrebbe essere ricalcolata e rivisualizzata ogni volta che viene modificato uno dei suoi vertici, il processo di elaborazione verrebbe rallentato notevolmente. Per questa ragione è prevista la funzione ENTUPD che permette di aggiornare sullo schermo anche una polilinea (o un blocco) modificata. ENTUPD è richiamata con il nome d'entità di un componente qualsiasi della polilinea o del blocco; non deve necessariamente trattarsi dell'entità d'intestazione, infatti ENTUPD la reperisce automaticamente. ENTUPD è prevista per polilinee e blocchi, ma puo essere usata anche per qualsiasi altra entità; essa rigenererà sempre l'entità sullo schermo, entità secondarie comprese.

Ad esempio, supponiamo che la prima entità di un disegno sia una polilinea con alcuni vertici:

```
(setq e1 (entnext))                      ; assegna alla polilinea il nome E1
(setq e2 (entnext e1))                   ; assegna al primo vertice il nome E2
(setq ed (entget e2))                    ; assegna ED al dato relativo al vertice
(setq ed
     (subst '(10 1.0 2.0)
               (assoc 10 ed)             ; cambia la posizione del vertice in ED
               ed                        : lo trasporta nel punto 1,2
     )
)
(entmod ed)                              ; muove il vertice nel disegno
(setupd e1)                              ; rigenera la polilinea E1
```

5.4.5 Restrizioni

I nomi di entità e i gruppi di selezione sono validi solo durante la sessione di editazione nella quale sono stati ottenuti da AutoCAD. Se si prova ad eseguire le seguenti operazioni mentre un comando come PLINEA o EDITATT è azionato, si otterrà un NIL e la funzione richiesta non verrà eseguita.

ENTMOD	-per modificare un'entità già esistente
ENTUPD	-per rigenerare un'entità complessa modificata
ENTDEL	-per ristabilire e rigenerare un'entità previamente cancellata

5.5 Uso di nomi d'entità e di gruppi di selezione in AutoCAD

I nomi d'entità e i gruppi di selezione sono risposte valide ai messaggi "Selezionare oggetti:" di AutoCAD che prevedono la selezione "Ultimo". Ne segue che le entità richiamate da LISP possono essere usate dai comandi di AutoCAD.

Nei casi in cui AutoCAD permette di selezionare oggetti tramite puntamento, il processo di selezione accetterà le liste del tipo di quelle fornite da ENTSEL. L'entità viene selezionata e viene specificato il punto di selezione. In questo modo LISP può trasmettere punti di selezione per un'entità specificata a comandi quali SPEZZA, RACCORDO, CIMA, TAGLIA ed ESTENDE. Occorre notare che le liste del tipo ENTSEL possono essere usate anche per altre selezioni, a condizione che il comando accetti la selezione tramite puntamento.

5.6 Note sulle curve del tipo V e sulle splinee

Quando utilizzate ENTNEXT per passare da un vertice all'altro di una polilinea, incontrerete probabilmente alcuni vertici che voi non avete esplicitamente creato. Infatti le opzioni "V" (curva a V) e "S" (splinea) del comando EDITPL creano automaticamente dei vertici ausiliari. Potete ignorarli, dal momento che ogni modifica apportata ad uno di questi vertici scomparirà la prossima volta che ricorrerete alle opzioni "V" e "S" di EDITPL con la polilinea in questione.

Dal gruppo d'entità con segnalatore 70 potete dedurre se la polilinea è stata trasformata in una curva a V o in una Splinea, nel primo caso il valore in bit sarà 2, nel secondo caso sarà 4. Se nessuno di questi due valori è presente, significa che i vertici della polilinea non sono stati manipolati. Se è presente il valore 2 (curva a V), un vertice su due avrà valore in bit uguale a 1 per indicare che si tratta di vertici aggiunti durante il procedimento per la produzione della curva a V. Se utilizzate ENTMOD per spostare i vertici con l'intento di modificare la curvatura richiamando EDITPL, vi conviene ignorare tali vertici.

Se è presente il valore 4 (splinea), troverete alcuni vertici con valore 1 (inseriti durante il procedimento di curvatura a V se la variabile SPINESEGS era negativa) e altri con valore 8 (inseriti durante la creazione della splinea), tutti gli altri vertici avranno valore 16 (punti di riferimento della splinea). Anche in questo caso, se utilizzate ENTMOD per spostare i vertici con l'intento di modificare la curvatura richiamando EDITPL, vi conviene ignorare tutti i vertici tranne quelli corrispondenti a punti di riferimento.

5.7 Accesso alla tavola dei simboli

Sono state inserite in AutoLISP le funzioni TBLNEXT e TBLSEARCH, che permettono l'accesso (per sola consultazione) alle tavole di simboli riguardanti i piani, i tipi di linea, gli stili di testo, la vista contrassegnata da nome e la definizione dei blocchi di AutoCAD. Diamo una breve descrizione di queste funzioni:

5.7.1 (tblnext < nome della tavola> [<primo>])

Questa funzione viene impiegata quando si vuole esaminare il contenuto di una intera tavola di simboli. Il primo argomento è una stringa che identifica la tavola di simboli che interessa. Nomi validi per la tavola sono: "LAYER", "LTYPE", "VIEW", "STYLE" e "BLOCK", "UCS" e "VPORT". Non c'è bisogno che la stringa sia impostata in lettere maiuscole. Se TBLNEXT viene usato ripetutamente, restituisce sempre la nuova entrata nella tavola specificata (la funzione TBLSEARCH, descritta in seguito, imposta la nuova entrata da richiamare). Se è presente anche

il secondo argomento e ha un valore diverso da NIL. la tavola dei simboli viene ricaricata e viene rintracciata la prima entrata alla tavola; altrimenti viene rintracciata la seconda entrata. Se non si riscontrano altre entrate, viene restituito NIL. Entrate cancellate non vengono mai restituite.

Se viene trovata un'entrata, viene ritornata sotto forma di lista di coppie puntate di valori e codici del tipo DXF, molto simili a quelle restituite da ENTGET. Ad esempio:

```
(tblnext "layer" T)          (rintracciare il primo piano)
```

può restituire:

```
((0 . "LAYER")               (tipo di simbolo)
 (2 . "0")                   (nome del simbolo)
 (70 . 0)                    (contrassegni)
 (62 . 7)                    (numero del colore. si ottiene un numero negativo
                              se il dispositivo colore non è azionato)
 (6 . "CONTINUOUS")          (nome del tipo di linea)
)
```

Va notato che qui non esistono gruppi "-1". AutoCAD ricorda l'ultima entrata che è stata restituita per ogni tavola e restituisce la successiva ogni volta che TBLNEXT viene chiamata per quella tavola. Quando si inizia a esaminare il contenuto di una tavola, è importante assicurarsi che il secondo argomento che si fornisce per ritornare alla prima entrata della tavola, non corrisponda a un valore NIL.

Entrate tratte dalla tavola "BLOCK" includono un gruppo "-2" che ha come "nome d'entità" la prima entità nella definizione di blocco (se ne esiste una). Quindi, dato un blocco chiamato "BOX":

```
(tblnext "block")            (rintracciare la definizione del blocco)
```

può restituire:

```
((0 . "BLOCK")                              (tipo di simbolo)
 (2. "BOX")                                 (nome del simbolo)
 (70 . 0)                                   (contrassegni)
 (10 9.0 2.0 0.0)                           (origini X.Y.Z)
 (-2 . <Nome dell'entità: 40000126>)        (prima entità)
)
```

Il nome d'entità nel gruppo "-2" è ammesso da ENTGET e da ENTNEXT, ma non da tutte le altre funzioni di accesso alle entità. Perciò non è possibile modificare una tale entità tramite ENTMOD e neppure usare SSADD o ENTSEL per porla in un gruppo di selezione. Fornendo a ENTNEXT il nome "-2" per un gruppo d'entità, è possibile esaminare tutte le entità comprese all'interno del blocco di definizione: ENTNEXT restituisce NIL dopo aver mostrato l'ultima entità nel blocco di definizione.

Se la variabile di sistema FLATLAND è 0, TBLNEXT restituirà sempre punti tridimensionali in forma di liste di 4 elementi del tipo (10 1.0 2.0 3.0). Se FLATLAND è diversa da 0, alcuni punti 3D vengono restituiti come liste a 3 elementi (2D) con la componente Z in coppie puntate separate di gruppo 30.

5.7.2 (tblsearch <nome della tavola> <simbolo>)[<prossimo>]

Questa funzione ricerca la tavola di simboli identificata col nome che viene inserito in <nome di tavola> (stessa procedura di TBLNEXT), e cerca il nome di simbolo che è stato inserito in <simbolo>. Entrambi i nomi inseriti vengono automaticamente convertiti in lettere maiuscole. Se viene trovata un' entrata per il nome di simbolo dato, tale entrata viene restituita nel formato descritto da TBLNEXT. Se invece non viene trovata nessuna entrata che corrisponda ai dati inseriti, appare NIL. Ad esempio.

```
(tblsearch "style" "standard")   (rintracciare lo stile di testo)
```

può dare come risposta:

```
((0 . "STYLE")          (tipo di simbolo)
 (2 . "STANDARD")       (nome del simbolo)
 (70 . 0)               (contrassegni)
 (40 . 0.0)             (altezza fissata)
 (41 . 1.0)             (fattore di profondità)
 (50 . 0.0)             (angolo obliquo)
 (71 . 0)               (contrassegni di generazione)
 (3 . "txt")            (file del carattere primario)
 (4 . "")               (file del carattere gigante)
)
```

Solitamente TBLSEARCH non ha nessun effetto sull'ordine delle entrate restituite da TBLNEXT. Però, se TBLSEARCH dà la risposta cercata e l'argomento <prossimo> è presente e ha valore diverso da NIL, il contatore d'entrate di TBLNEXT viene impostato in modo tale che la prossima chiamata di TBLSEARCH restituirà l'entrata seguente a quella restituita nella presente chiamata.

L'argomento <prossimo> è molto utile quando si lavora con la tavola di simboli VPORT dato che tutte le finestre in una configurazione particolare portano il medesimo nome. Per trovare ed elaborare ogni finestra nella configurazione chiamata "4VIEW", ad esempio, potete utilizzare il codice seguente:

```
(setq v (tlbsearch "VPORT" "4VIEW" T))    ; trova la prima entrata
(while (and v (= (cdr (assoc 2 v))  "4VIEW"))
  ... elabora l'entrata ...
  (setq v (tblnext "VPORT"))              ; trova la prossima entrata
)
```

Se la variabile di sistema FLATLAND è 0, TBLSEARCH restituirà sempre punti tridimensionali in forma di liste di 4 elementi del tipo (10 1.0 2.0 3.0). Se FLATLAND è diversa da 0, alcuni punti 3D vengono restituiti come liste a 3 elementi (2D) con la componente Z in coppie puntate separate di gruppo 30.

5.8 Accesso allo schermo grafico e ai dispositivi di input

Le funzioni descritte in questo paragrafo permettono di accedere direttamente allo schermo grafico di AutoCAD e ai dispositivi di input a partire da LISP e permettono a comandi implementati in LISP di interagire con l'utente come se fossero stati implementati in AutoCAD. Queste funzioni possono provocare disturbi sullo schermo; la sequenza che segue serve a eliminare questi inconvenienti:

(grtext)
(redraw)

Queste funzioni sono destinate *solo a programmatori esperti*. Comunque, la maggior parte delle applicazioni di LISP non le utilizzano. A coloro che intendono servirsi di queste funzioni occorre far notare che la loro azione può cambiare da una versione di AutoCAD all'altra e che la Autodesk AG non garantisce la compatibilità verso l'alto delle applicazioni che fanno uso di queste funzioni. Inoltre, è probabile che le applicazioni che utilizzano le funzioni GRTEXT e GRREAD non funzionino in maniera identica su configurazioni hardware diverse, a meno che il programmatore si attenga alle regole d'uso descritte qui di seguito.

5.8.1 (grclear)

Questa funzione cancella la finestra corrente. (Sui sistemi monoschermo passerà prima dallo schermo testo allo schermo grafico.) Le aree riservate alle sequenze di comando, ai menu e alle informazioni correnti, restano invariate. Per ripristinare l'immagine sullo schermo grafico basta usare la funzione REDRAW.

5.8.2 (grdraw <da> <a> <colore> [<evidenziare>])

GRDRAW disegna un vettore fra due punti nella finestra corrente. <da> e <a> sono i punti bidimensionali o tridimensionali (liste di due o tre numeri reali) che specificano le estremità del vettore relativamente all'UCS corrente. AutoCAD ritaglia il vettore in modo da adeguarlo allo schermo. Il vettore viene disegnato con il colore specificato dall'argomento <colore> ~~con -1 che~~ significa "inchiostro XOR", cioè un inchiostro immaginario che cancella elementi disegnati quando li incontra e viene cancellato a sua volta se gli si disegna sopra. Se è presente l'argomento opzionale <evidenziare> e questo è diverso da zero, il vettore viene evidenziato (solitamente con linea tratteggiata). Se <evidenziare> non è fornito o è pari a zero, viene usato il modo di visualizzazione normale.

5.8.3 (grtext [<casella> <testo> [<evidenziare>]])

GRTEXT permette ad AutoLISP di scrivere nelle porzioni di testo dello schermo grafico di AutoCAD. Se richiamato con un numero di casella tra 0 e la casella del menù di schermo con il numero più alto meno 1, GRTEXT visualizza l'argomento di stringa <testo> nella casella specificata. Il <testo> viene troncato se oltrepassa la lunghezza della casella oppure gli vengono aggiunti spazi vuoti quando non la riempie. Se l'argomento opzionale <evidenziare> è presente ed è diverso da zero, il testo nella casella indicata verrà visualizzato normalmente (si noti che evidenziare una casella ha come conseguenza che una qualsiasi altra casella evidenziata viene visualizzata normalmente). Quando si scrive nelle caselle di menù, il testo deve essere scritto dapprima senza l'argomento <evidenziare> e in seguito evidenziato. Ai richiami di visualizzazione normale e di evidenziazione bisogna fornire la stessa stringa di testo (quella scritta nella casella). Se non si osservano queste regole, è probabile che i programmi in LISP funzionino diversamente su dispositivi di visualizzazione diversi.

Occorre notare infine, che questa funzione visualizza il testo fornito nell'area di menù sullo schermo senza modificare l'opzione di menù corrispondente. Inoltre, su alcuni monitor, le voci del menù normale vengono evidenziate ricolorando il testo della voce, per cui il testo GRTEXT ritornerà al testo della voce di menu quando l'utente evidenzia la porzione dello schermo occupata da questo. Su altri monitor, l'area di menù viene riscritta ogni volta che viene eseguita

un'operazione di cambio schermo o ogni volta che lo schermo viene rigenerato o ridisegnato. Sulla maggior parte dei monitor, comunque, il testo GRTEXT rimarrà nell'area di menù finchè il menù non viene sostituito da un altro menù.

Se GRTEXT è richiamata con il numero di casella -1, il testo verrà scritto sulla riga monitor di stato e di modo. La lunghezza di questa riga differisce da un monitor all'altro (la maggior parte dei monitor accetta 40 caratteri, un'eccezione è il Color Graphics Adaptor di IBM). Se il testo non rientra nello spazio a disposizione, esso verrà troncato.

Se si usa il numero di casella -2, GRTEXT scriverà il testo nella riga monitor delle coordinate. Si noti che se è attivata la visualizzazione continua delle coordinate, i valori scritti in questa casella verranno sostituiti non appena il puntatore invia una nuova coppia di coordinate. Per i numeri di casella -1 e -2 l'argomento <evidenziare> non viene preso in considerazione.

Infine, GRTEXT può essere richiamata senza argomenti per ridare i valori standard a tutte le aree di testo sullo schermo.

5.8.4 (grread [<aggiornare>])

GRREAD permette di leggere direttamente i dati immessi dai dispositivi di input di AutoCAD e c'è la possibilità di associare la visualizzazione delle coordinate agli spostamenti del puntatore. Solo comandi molto specializzati usano questa funzione, infatti la maggior parte delle immissioni in AutoLISP sono effettuate tramite le varie funzioni "GETxxx" quali GETSTRING, GETREAL e cosi via. L'argomento <aggiornare>, se è presente e diverso da NIL, attiva la visualizzazione continua delle coordinate fornite dal puntatore in movimento e non richiede una selezione tramite pulsante di selezione. Questo è il meccanismo che AutoCAD utilizza per la specificazione dinamica ("trascina").

Questa funzione restituisce una lista il cui primo elemento è un codice che specifica il tipo di immissione. Il secondo elemento della lista è un numero intero o una lista che rappresenta un punto, a seconda del tipo di immissione. I codici per il primo elemento della lista sono:

2	Carattere di tastiera - codice ASCII come secondo elemento
3	Punto selezionato - coordinate sottoforma di lista
4	Casella di menu di schermo selezionata - numero di casella come secondo elemento
5	Coordinate del modo Trascina come secondo elemento: restituite solo se sono il secondo elemento e se diverse da NIL.
6	Voce selezionata del menù BUTTONS -numero del pulsante come secondo elemento
7	Voce selezionata del menù TABLET1 -numero della casella come secondo elemento
8	Voce selezionata del menù TABLET2 -numero della casella come secondo elemento
9	Voce selezionata del menù TABLET3 -numero della casella come secondo elemento
10	Voce selezionata del menù TABLET4 -numero della casella come secondo elemento
11	Voce selezionata del menù AUX1 -numero della casella come secondo elemento
12	Coordinate associate al pulsante del puntatore restituite come secondo elemento. Segue sempre una lista restituita del tipo 6.
13	Selezione di una voce del menù di schermo evidenziata tramite immissione da tastiera. Il numero della casella è restituito come secondo elemento.

Se il secondo elemento nella lista restituita è un punto, l'impostazione di FLATLAND determina se il punto sarà 3D o 2D. Se FLATLAND è = 0, il punto sarà un punto 3D relativo all'UCS corrente, altrimenti sarà un punto 2D.

Se si preme CTRL C mentre la funzione GRREAD è in corso, si interrompe il programma LISP. Ogni immissione ulteriore verrà trasmessa direttamente a GRREAD.

Capitolo 6

Gestione della memoria

I simboli, le funzioni definite dall'utente e le funzioni predefinite descritte in questo manuale sono memorizzate nella memoria del computer soltanto per la durata della sessione di editazione in AutoCAD. Quando si lancia AutoLISP, esso richiede due aree di memoria di dimensioni consistenti; la prima, chiamata "heap", è l'area in cui sono memorizzate tutte le funzioni e le variabili (chiamate anche *nodi*): più queste aumentano (e più le funzioni sono complicate), più spazio verrà occupato nella memoria di "heap". La seconda area di memoria, chiamata "stack", contiene argomenti di funzioni e risultati parziali di funzioni; più aumentano i livelli delle funzioni o le ricorsioni, più spazio verrà occupato nella memoria di "stack".

6.1 Adattare la memoria alle esigenze di AutoLISP

Questa sezione si riferisce solo ai sistemi PC-DOS/MS-DOS che gestiscono AutoLISP normale oppure AutoLISP Esteso. In altri ambienti, i programmi AutoLISP e i dati memorizzati possono essere di dimensioni illimitate.

I valori standard per le aree di "heap" e "stack" sono:

Heap = 40'000 bytes (per AutoLISP normale)
Stack = 3'000 bytes

Questi valori si sono rivelati come i più idonei per la maggior parte delle applicazioni. (le dimensioni della memoria di "heap" in AutoLISP Esteso sono determinate dalla quantità di memoria estesa assegnata a quest'ultimo).

AutoLISP non può espandere la sua memoria di "heap" o di "stack" mentre è lanciato. Se il numero di funzioni e variabili definite è così grande da occupare tutta la memoria di "heap", AutoLISP visualizza il messaggio:

Insufficient node space (Spazio nodale insufficiente) *oppure*
Insufficient string space (Spazio di stringa insufficiente)

e verrà interrotta l'esecuzione della funzione in corso. Nel caso non ci fosse memoria sufficiente per caricare AutoLISP quando si esegue AutoCAD, appare il messaggio:

Insufficient memory -- AutoLISP disabled
(Memoria insufficiente - AutoLISP disattivato)

Per potere utilizzare AutoLISP occorre mettere a disposizione una maggiore quantità di memoria ed eseguire nuovamente AutoCAD.

Se il concetto di aree di "heap" e "stack" così come sono usate nei linguaggi di programmazione vi è familiare, avete la possibilità di ricorrere al comando "SET" del sistema operativo per modificare la quantità di memoria dedicata a queste aree in AutoLISP. Ad esempio, i comandi:

```
C>SET  LISPHEAP=30000
C>SET  LISPSTACK=10000
```

inducono AutoLISP a riservare 30'000 bytes di memoria per l'area di "heap" e 10'000 bytes per l'area di "stack". Userete questi valori se i vostri programmi fanno un uso frequente della ricorsione (che richiede ulteriore spazio di stack) ma non abbisognano di altrettanto spazio di memoria di heap per funzioni e simboli. La somma delle due aree non può oltrepassare 45'000 bytes. Istruzioni di questo tipo possono essere incluse nel file "autoexec.bat" in modo che vengano eseguite ogni volta che si fa il "bootstrap" del computer. Questi comandi SET hanno effetto solo su AutoLISP e non alterano la disponibilità di memoria quando AutoLISP non è lanciato. AutoLISP Esteso ignora l'impostazione di LISPHEAP.

Per AutoLISP Esteso, la variabile d'ambiente LISPXMEM specifica la posizione e le dimensioni dell'area dellla memoria estesa nella quale risiede AutoLISP e che verrà utilizzata come spazio di heap e di stack. Un valore adeguato a questa variabile permetterà ad AutoLISP Esteso di coesistere con altri programmi (compreso AutoCAD) che utilizzano una porzione di questa memoria. Se non viene fornito nessun valore per LISPXMEM, AutoLISP Esteso utilizzerà tutta la memoria estesa non sfruttata dal programma di gestione del disco di RAM standard (VDISK). AutoLISP Esteso fa in modo che l'area di heap e l'area di stack sommate corrispondano a 14 megabytes se è disponibile una quantità di memoria estesa sufficiente.

LISPXMEM viene impostata servendosi del comando di DOS "SET" prima di lanciare EXTLISP. Potete utilizzare per impostare LISPXMEM uno dei formati di comando seguenti.

1. C> SET LISPXMEM=*partenza*
2. C> SET LISPXMEM=*partenza,dimensione*
3. C> SET LISPXMEM=*,dimensione*

Il primo formato specifica l'indirizzo di partenza più basso che può essere utilizzato per AutoLISP Esteso. Questo valore deve essere tra 0100000 e 01000000 (esadecimali) o tra 1024K e 1638K (espressi in kilobytes). In altre parole, il valore deve essere espresso in termini di indirizzo assoluto oppure in kilobytes. Per specificare in kilobytes occorre aggiungere una "K" al numero (es. 1024K = 0100000). Il valore viene considerato come decimale a meno che non inizi per "0" o "0x", nel qual caso sarà esadecimale. Questo valore definisce solamente una zona minima nell'area di memoria estesa che AutoLISP Esteso prenderà in considerazione. Se, ad esempio, viene specificato un indirizzo di partenza che si trova a metà del buffer del VDISK, l'area assegnata ad AutoLISP Esteso comincerà alla fine del buffer del VDISK.

Il secondo formato specifica invece una zona di memoria estesa fornendo l'indirizzo di partenza e un dato spazio (dimensione) al quale AutoLISP Esteso dovrà limitarsi. Gli argomenti di partenza e dimensione devono essere dello stesso genere e trovarsi all'interno dell'intervallo 0100000 (1024K) - 01000000 (1638K). Ad esempio, per fare in modo che AutoLISP Esteso faccia uso di 1 megabyte di memoria estesa partendo da 1664K (lasciando quindi i primi 640 kilobytes di memoria estesa a disposizione di un'altro programma), potete servirvi di uno dei due formati seguenti:

C> SET LISPXMEM=0x1a0000,0x100000
C> SET LISPXMEM=1664K,1024K

il terzo formato specifica soltanto la quantità massima di memoria utilizzabile da AutoLISP Esteso e lascia ad AutoLISP Esteso di decidere il proprio indirizzo di partenza nella maniera usuale.

Se esiste una variabile d'ambiente LISPXMEM ma non è conforme a uno di questi tre formati, AutoLISP Esteso utilizzerà la sua impostazione standard, occuperà cioè tutta la memoria estesa.

LISPXMEM è molto simile sia per formato che per funzione alla variabile d'ambiente ACADXMEM di AutoCAD. L'unica eccezione è che ACADXMEM dispone di un quarto formato possibile (="*niente*"), che non è impiegabile con LISPXMEM e inoltre ACADXMEM serve a controllare lo spazio di memoria di impaginazione I/O di AutoCAD mentre LISPXMEM controlla l'uso che AutoLISP Esteso fa della memoria estesa. Bisogna fare attenzione a fornire alle due variabili valori che non si sovrappongano, ad esempio:

C> SET LISPXMEM=4096K,2048K
C> SET ACADXMEM=6144K

Partendo dal presupposto che il vostro computer dispone di almeno 5 megabytes di memoria estesa installata, questa impostazione lascerebbe 2 megabytes a disposizione di AutoLISP Esteso e i primi 3 megabytes liberi ad uso di un altro programma o di un disco di RAM. L'area di memoria estesa di impaginazione I/O di AutoCAD seguirebbe AutoLISP Esteso e utilizzerebbe il resto della memoria estesa (ma non più di 4 megabytes, il limite attuale di AutoCAD).

6.2 Ricupero di memoria nodale

Quando si creano funzioni e variabili destinate a un impiego temporaneo, si può associarvi NIL per eliminarle a uso ultimato. Supponendo di aver creato una funzione chiamata IMPOST che non serve più, si può digitare:

```
(setq impost nil)
```

per sbarazzarsene. La memoria nodale (heap) utilizzata dalla funzione viene ricuperata rendendola disponibile per altre funzioni e simboli.

Se si vuole liberare la memoria nodale utilizzata da una variabile del tipo descrittore file, restituita dalla funzione OPEN, è necessario eseguire un CLOSE prima di impostare NIL. Esiste solo un numero limitato di spazi disponibili per files aperti e se si dimentica di chiudere un file questo può fare in modo che lo spazio rimanga occupato, limitando quindi il numero di nuovi files che possono essere aperti.

Esiste anche la possibilità di eliminare *tutte* le funzioni e variabili caricate o definite durante una sessione di editazione. AutoLISP gestisce una lista chiamata ATOMLIST che contiene inizialmente i nomi di tutte le funzioni e variabili predefinite. (Per esaminarne il contenuto basta digitare "!atomlist" in risposta al messaggio "Comando:" di AutoCAD.) Quando si creano nuove funzioni e variabili, i loro nomi vengono aggiunti in testa alla lista ATOMLIST, quindi per eliminare tutto ciò che è stato definito si può ridurre ATOMLIST alla sua dimensione originale; attenzione tuttavia a non eliminare le funzioni predefinite del sistema.

Aggiungendo la funzione seguente al file "acad.lsp":

```
(defun C:ELIM (/ i item)
     (setq i 0)
     (while (not(equal(setq item (nth i atomlist)) 'C:ELIM))
         (if (= (type (eval item)) 'FILE)   ;assicurarsi che
i files siano chiusi
              (close (eval item)))
          (setq i (1+ i))
     )
     (setq atomlist (member 'C:ELIM atomlist)
     'ESEGUITO
```

)

si può richiamare il comando ELIM per ridurre la lista ATOMLIST e ricuperare la memoria nodale usata dalle nuove funzioni e variabili. Se C:ELIM è l'*ultima* funzione nel file "acad.lsp", le funzioni definite precedentemente nel file non verranno eliminate da ELIM dato che sono trattate come funzioni predefinite.

Il meccanismo di ATOMLIST non è una caratteristica standard di LISP e può essere soggetta a modifiche nelle versioni future di AutoLISP. Inoltre non è possibile utilizzare la tecnica di riduzione della ATOMLIST se la paginazione virtuale delle funzioni è attivata; vedi paragrafo che segue.

6.3 Paginazione virtuale di funzioni

Se un'applicazione di AutoLISP oltrepassa la memoria nodale disponibile (determinata dalla variabile d'ambiente LISPHEAP o dalla porzione di memoria estesa accessibile ad AutoLISP Esteso), si può attivare la *paginazione virtuale di funzioni* di AutoLISP per permettere al programma di crescere; occorre eseguire la funzione:

```
(vmon)
```

prima del primo DEFUN nel programma. Questo attiverà la funzione di paginazione virtuale per la durata della sessione di editazione in AutoCAD. Una volta attivata, la funzione di paginazione non può più essere disattivata. Solo funzioni create con un DEFUN successivo al richiamo di VMON verranno impaginate mentre funzioni definite prima del richiamo di VMON non verranno trasferite nella memoria di massa e continueranno a causare interruzioni per "insufficient node space".

Dopo l'esecuzione della funzione VMON, AutoLISP trasferisce nella memoria di massa le funzioni usate raramente ogni volta che si supera lo spazio nodale a disposizione e le richiama automaticamente quando sono richieste dall'utente. La paginazione avviene in modo automatico ed è del tutto trasparente. Le funzioni sono trasferite in un file temporaneo, gestito dalla paginazione di file di AutoCAD. Ciò significa che se l'utente dispone di espansione di memoria o di memoria estesa sufficiente, essa verrà usata per l'archiviazione temporanea delle funzioni e ne risulterà un'operazione molto più rapida che un trasferimento su disco.

Bisogna tenere presente che questa tecnica di memoria virtuale pagina solo *funzioni*; ciò significa che occorre avere ancora abbastanza memoria nodale per l'archiviazione di tutte le liste di dati dei programmi nonché i nomi delle funzioni e delle variabili. Quindi, anche se la paginazione permette di scrivere programmi molto lunghi senza incorrere in problemi di memoria, bisogna comunque determinare la dimensione ottimale per la memoria di "heap" mediante SET LISPHEAP. La paginazione virtuale di funzioni non richiede modifiche nell'impostazione di LISPSTACK.

6.4 Considerazioni tecniche

Le osservazioni seguenti riguardano meccanismi interni ad AutoLISP che possono essere soggetti a modifiche senza previa informazione. Tali osservazioni sono destinate a programmatori pratici di LISP. I principianti possono (e dovrebbero) ignorarle.

6.4.1 Spazio nodale

Un *nodo* è una unità di memoria in grado di rappresentare tutti i tipi di dati di AutoLISP.Di solito nei sistemi PC-DOS/MS-DOS. un nodo è lungo 10 bytes. Sotto OS/2 e sulle stazioni di lavoro a 32 bit, un nodo è pari a 12 bytes. AutoLISP Esteso utilizza nodi di 12 bytes. Per evitare frammentazioni di memoria e un sovraccarico della memoria di heap, i nodi vengono posizionati dalle heap in gruppi chiamati *segmenti*. La dimensione standard di un segmento corrisponde a 512 nodi (5120 bytes per nodi di 10 bytes).

AutoLISP dispone di una lista aggiornata dei nodi "liberi" (nodi al momento non associati ad un simbolo). Quando AutoLISP necessita di un nodo per archiviare un simbolo o un valore. AutoLISP consulta la lista dei nodi liberi per reperire un nodo disponibile. Se non ne trova, esegue un riordinamento automatico (garbage collection) posizionando i nodi che non sono più associati ad un simbolo nella lista dei nodi liberi. Uno di questi nodi viene quindi scelto. Se tramite il riordinamento si ottengono troppo pochi nodi liberi, AutoLISP prende un segmento addizionale dalla heap. Se vengono invece trovati un numero sufficiente di nodi liberi, questi vengono posizionati nella lista o uno di loro viene scelto per soddisfare alla richiesta del momento. Se non e possibile reperire nessun nodo libero dalla heap, viene richiamata la funzione di paginazione virtuale (se attiva) per liberare alcuni nodi trasferendo sul supporto di memoria ausiliaria la funzione utilizzata più recentemente. Altrimenti viene visualizzato il messaggio "insufficient node space" e la funzione richiedente lo spazio nodale viene interrotta. E' importante notare che lo spazio nodale non viene *mai* restituito alla heap finchè non si abbandona AutoCAD.

E' possibile forzare un'operazione di riordino servendosi della funzione GC:

```
(gc)
```

Va comunque notato che le operazioni di riordino sono molto dispendiose in termini di tempo e non dovrebbero essere eseguite se non in casi strettamente necessari. Conviene lasciare ad AutoLISP di decidere quando eseguire un tale riordino.

6.4.2 Spazio di stringa

Lo spazio di memoria di stringa viene sottratto dalla memoria di heap così come i segmenti nodali. Se il vostro programma richiama le funzioni di allocamento automatico (descritte in seguito) per assegnare tutta la memoria disponibile a nodi, è molto probabile che otterrete un messaggio "insufficient string space" con effetti altrettanto nefasti del messaggio "insufficient node space". Conviene lasciare alla funzione di allocamento automatico di fare in modo che lo spazio di stringa resti sufficiente. Lo spazio di stringa è impiegato per archiviare da nomi di simboli a messaggi di richiesta o stringhe di menù, tutte voci che vengono sottoposte ad AutoLISP per una valutazione. Se vi servite di voci di menù molto lunghe con espressioni di valutazione di AutoLISP, queste richiederanno dalla heap una quantità significativa di spazio di stringa contiguo.

6.4.3 Archiviazione di simboli

La struttura della memoria in AutoLISP è basata su un sistema di puntatori. L'utilizzo di nodi è onnipresente, tutto viene rappresentato tramite una struttura nodale. Il semplice atto di far corrispondere un simbolo ad un valore come in:

```
(setq longsym 3.1415)
```

richiede l'impiego di tre nodi: uno per il posizionamento del simbolo all'interno di ATOMLIST, uno per memorizzare il nome del simbolo e uno per memorizzare il suo valore. Se il nome del simbolo è lungo al massimo sei caratteri, questo viene memorizzato direttamente nel nodo del nome del simbolo, altrimenti gli viene assegnato uno spazio di stringa dalla heap per memorizzare il nome, il nodo del nome del simbolo fa riferimento quindi a questa stringa. Utilizzando nomi brevi per simboli si ha perciò il vantaggio di ridurre lo spazio occupato ed evitare frammentazioni della heap.

6.4.4 Allocazione manuale

Potete utilizzare le funzioni ALLOC e EXPAND per controllare manualmente lo spazio di allocazione di nodo e di stringa e perfezionare l'efficienza delle vostre applicazioni. Utilizzando queste espressioni all'inizio del vostro file "acad.lsp" potete preallocare i nodi e riservare una certa quantità di spazio di stringa. Questo può ridurre il numero di riordini e aumentare la velocità dell'applicazione.

Potete usare la funzione ALLOC per modificare le dimensioni dei segmenti (il valore standard corrisponde a 512 nodi ciascuno).

```
(alloc <numero>)
```

ALLOC assegna ad un segmento la quantità di nodi corrispondente a <numero> e restituisce il valore precedente.

Utilizzando la funzione EXPAND, potete allocare manualmente lo spazio nodale impostando un numero di segmenti.

```
(expand <numero>)
```

<numero> è il numero di segmenti che si desidera allocare. EXPAND restituisce il numero di segmenti che si possono prendere dalla heap. Tale valore può essere molto inferiore al numero impostato dal momento che dipende dalla quantità di heap ancora libera.

Facciamo alcuni esempi riguardanti AutoLISP normale sotto PC-DOS/MS-DOS (nodi a 10 bytes) dato che in questo ambiente lo spazio nodale è molto importante.

```
(alloc 1024)
```

assegna ad un segmento le dimensioni di 1024 nodi, ciò significa che lo spazio di heap necessario dovrà essere pari a 10240 bytes per segmento.

```
(alloc 100)
```

assegna ad un segmento valore 100, lo spazio di heap necessario sarà quindi di soli 1000 bytes per segmento.

Se le dimensioni di un segmento vengono mantenute al loro valore standard di 512 nodi, una impostazione a

```
(expand 10)
```

richiederà 10 segmenti (51200 bytes), cioè più di quanto è disponibile sotto DOS per uso con AutoLISP normale. Si otterranno quindi solo 1 o 2 segmenti, ma chiederne di più non danneggia il sistema.

Vediamo ora questo esempio pratico:

```
(alloc 3000)     ; un segmento sarà pari a 3000 nodi
(expand 1)       ; si ottengono 3000 nodi liberi (cioè un segmento)
(alloc 10000)    ; aumenta le dimensioni di un segmento per evitare di
                 richiedere ulteriori segmenti
```

In questo caso vengono utilizzati 30000 bytes per spazio nodale e il resto della heap viene destinato come spazio di stringa. I nodi verranno preallocati e posizionati nella lista dei nodi liberi. Nessuna operazione di riordino sarà necessaria finchè non sono stati utilizzati tutti i 3000 nodi. Dopo che questi nodi sono stati esauriti si rivelerà necessaria un'operazione di riordino per recuperare ulteriore spazio. (alloc 10000) assegna ad un segmento un valore che evita di allocare ulteriori segmenti dalla heap, il cui spazio rimanente resterà consacrato alle stringhe.

Se faceste il contrario, cioè diminuiste successivamente le dimensioni di un segmento finchè la richiesta di un nodo si riveli insoddisfacente (es. "(alloc 1) (expand 1)" restituisce 0) finireste con l'utilizzare tutta la heap per spazio nodale e ottenere il messaggio di errore "insufficient string space" e in assenza di spazio per stringhe AutoLISP non potrebbe più essere utilizzato.

6.4.5 Statistiche di memoria

La funzione MEM:

```
(mem)
```

visualizza lo stato corrente della memoria di AutoLISP e restituisce NIL. "Nodes:" è il numero totale di nodi allocato. Tale valore dovrebbe corrispondere alle dimensioni di un segmento nodale moltiplicato per il numero di segmenti. "Free nodes:" è il numero di nodi compreso nella lista dei nodi liberi e individuato da un'operazione di riordino. "Segments:" è il numero di segmenti nodali allocato e la voce "Allocate:" corrisponde alle dimensioni correnti di un segmento. "Collections" riporta il numero di operazioni di riordino eseguite, automatica o richieste dall'utente.

Se VMON è attivata vengono aggiunti due ulteriori voci "Swap-in" che riporta il numero di funzioni che sono state immagazzinate in memoria dal file di pagina creato dal sistema di memoria virtuale. Si tratta di funzioni di richiesta di impaginazione introdotte dal file di pagina sollecitato da un semplice richiamo di funzione. La voce "Page file:" riporta le dimensioni del file di pagina creato per raccogliere le funzioni trasferite alla memoria esterna.

6.4.6 Paginazione virtuale di funzioni

Dopo che è stata eseguita la funzione VMON, tutte le funzioni DEFUN collocano un nuovo nodo chiamato *tabella di paginazione* all'inizio di ogni lista di funzione. Questo nodo è aggiunto prima della lista che contiene gli argomenti formali. I nodi delle tabelle di paginazione devono essere utilizzati esclusivamente per la paginazione e in nessun caso per programmi scritti dall'utente. La funzione TYPE restituisce PAGETB per questi nodi.

Quando AutoLISP non ha più abbastanza nodi a disposizione, trasferisce la funzione meno richiesta nel file di paginazione, salva il suo indirizzo nella tabella di paginazione e libera tutti i nodi della funzione che segue la tabella di paginazione. La tabella di paginazione è marcata per indicare che la funzione è stata trasferita. Quando una funzione trasferita viene valutata, essa viene caricata dal file di paginazione (è possibile che a questo punto altre funzioni vengano trasferite nella memoria ausiliaria) prima dell'esecuzione. Basta che una funzione sia stata trasferita nel file di paginazione una volta; ulteriori trasferimenti liberano semplicemente i suoi nodi; non occorre digitare la funzione dato che è già presente nel file.

In AutoLISP le funzioni create con DEFUN sono semplicemente delle liste e possono essere manipolate come qualsiasi altra lista. I programmi che lo fanno devono tener conto della paginazione (o prescindere da VMON). Le funzioni create con DEFUN hanno come primo nodo la tabella di paginazione, che quindi si deve tralasciare se si esamina la funzione. Se si crea una funzione come lista (senza DEFUN), essa funzionerà perfettamente ma non sarà soggetta alla paginazione: ciò significa che si può facilmente oltrepassare la memoria disponibile se si ricorre spesso a questo metodo. D'altra parte è possibile bloccare una funzione in memoria ridefinendola senza la sua tabella di paginazione. Ad esempio, per bloccare in memoria una funzione di nome ZORP definita con DEFUN, si potrebbe digitare:

```
(setq zorp (cdr zorp))
```

per cancellare la tabella di paginazione iniziale. Le tabelle di paginazione sono rappresentate da spazi vuoti quando si stampa la funzione con PRINT. Per stabilire se una funzione è trasferibile o meno basta verificare se dopo la prima parentesi di sinistra c'è uno spazio; la presenza di uno spazio indica che è trasferibile.

Se si tenta di localizzare una funzione come dato e questa è stata trasferita nella memoria ausiliaria, si troverà solo la tabella di paginazione nella lista delle funzioni. L'accesso alla funzione non basterà per caricarla, solo la sua *valutazione* avrà questo effetto. Quindi se si ha l'intenzione di costruire funzioni e modificarle direttamente, conviene costruirle sottoforma di liste invece di usare DEFUN, oppure ricorrere all'espediente descritto sopra per bloccarle nella memoria.

La lista di simboli riconosciuti da AutoLISP si chiama ATOMLIST. Un programma sofisticato può manipolare questa lista per ricuperare spazio o per localizzare tutti i simboli. (Come già osservato nel paragrafo precedente, questa non è una caratteristica standard di LISP e può essere soggetta a modifiche in versioni future di AutoLISP.) Quando una funzione è trasferita nella memoria ausiliaria la sua codificazione viene effettuata in base alla supposizione che ATOMLIST non cambierà durante la paginazione (se la paginazione funzionasse in questo modo, richiederebbe molto più spazio su disco e sarebbe cinque volte più lenta). Per rendere caricamento di file ulteriormente efficiente, i simboli contenuti in ATOMLIST vengono riordinati dinamicamente (i simboli richiamati più recentemente vengono posti in cima alla lista) se viene azionata la funzione di paginazione virtuale. Per garantire che ATOMLIST non cambi, ATOMLIST viene tolta dalla lista dei simboli accessibili dall'utente. Se si utilizza la paginazione virtuale di funzioni, non si deve manipolare ATOMLIST in nessun caso!

gruppo di facce 3D, ciò permette l'editazione di ogni oggetto come entità unica (per trasformare questi oggetti in gruppi di facce 3D basta applicare il comando ESPLODE).

Potete utilizzare questo programma in diversi modi:

- Se selezionate la voce "Costruzioni 3D" dal menù a rotolo "Disegno", appare un menù ad icone dal quale si possono selezionare gli oggetti tridimensionali implementati in questo programma. La prima volta che si compie una selezione, AutoCAD carica il programma, se non è già stato caricato.

- Se selezionate "Costruzioni 3D" dal sottomenù di schermo "3D", AutoCAD carica il programma (se non è già stato caricato) ed è possible selezionare gli oggetti desiderati dal menù di schermo.

- E' possibile anche caricare il programma direttamente digitando:

  Comando: <u>(load "3d")</u>

  Una volta caricato il programma, si puo immettere "3D" in risposta al messaggio "Comando:", il programma reagisce presentando una lista degli oggetti 3D disponibili.

  Comando: <u>3D</u>
  SCatola/Cono/Scodella/CuPola/Rete/Piramide/SFera/Toro/CUneo:

  "3d.lsp" definisce una funzione "C:XXX" per ognuno dei tipi di oggetti, potete quindi immettere il nome dell'oggetto direttamente in risposta al messaggio "Comando:". Ad esempio

  Comando: <u>TORO</u>

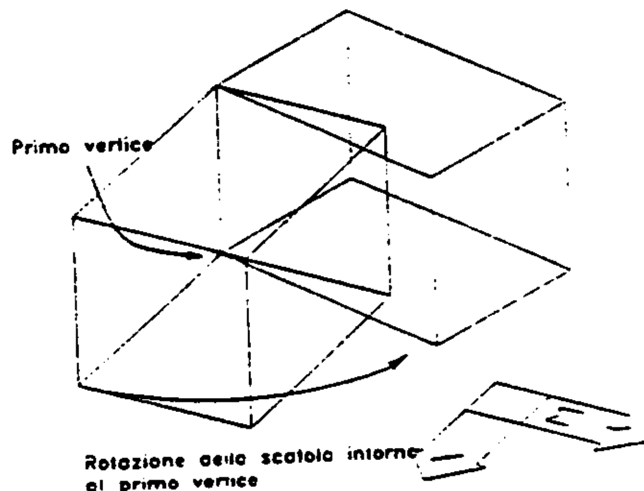
  è sufficiente affinchè AutoCAD inizi a costruire un toro.

Le singole opzioni del comando sono descritte qui di seguito.

A.3.1.1 SCATOLA - Scatola o cubo 3D

Questo comando serve a disegnare scatole a forma di parallelepipedo o di cubo. Si inizia designando il primo vertice della scatola. La lunghezza, larghezza, altezza e angolo di rotazione della scatola verranno specificate relativamente a questo vertice.

Comando: SCATOLA
Vertice della scatola: *immettere punto*
Lunghezza: *immettere distanza*
Cubo/<Larghezza>: *distanza o "C"*



Se rispondete alla richiesta Cubo/<Larghezza> immettendo una seconda distanza, AutoCAD la utilizza come valore della larghezza della scatola e domanda il valore dell'altezza. Se invece si immette "C" o "Cubo", AutoCAD disegna una scatola cubica, utilizzando lo stesso valore della lunghezza anche per la larghezza e per l'altezza. In entrambi i casi il messaggio di richiesta seguente sarà:

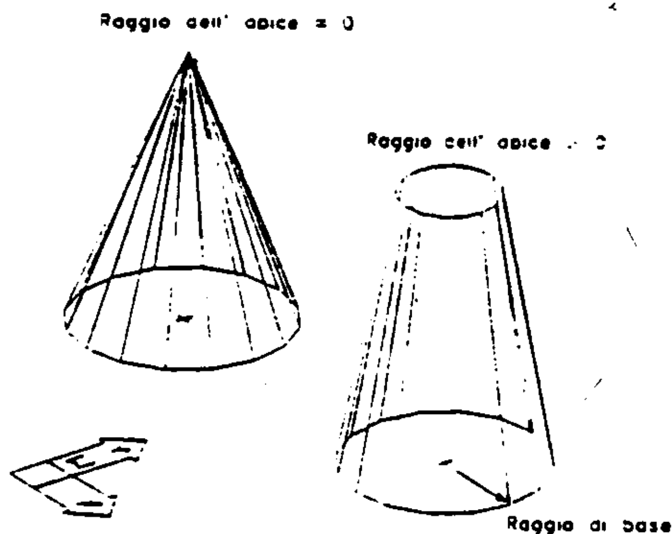
Angolo di rotazione intorno all'asse Z: *immettere un angolo*

L'angolo di rotazione specificato è relativo all'asse X corrente; il punto di base per la rotazione è il primo vertice della scatola. L'asse di rotazione è parallelo all'asse Z corrente.

A.3.1.2 CONO

Questo comando costruisce un cono dopo che ne sono stati specificati raggio e diametro della base e dell'apice e l'altezza della figura.

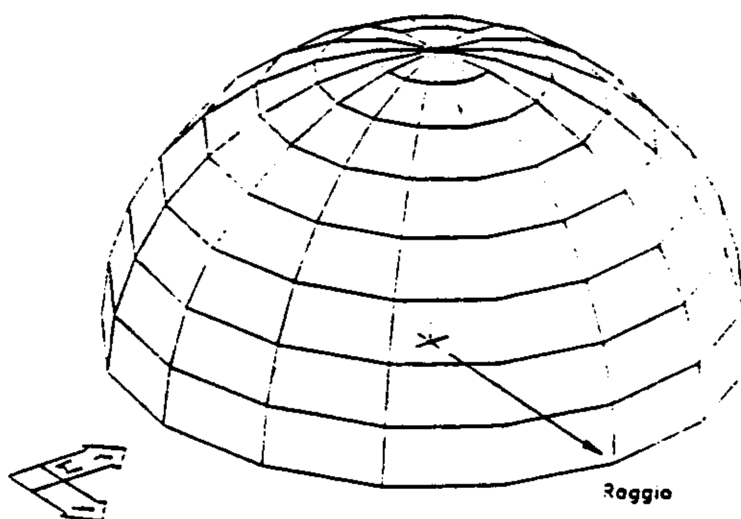
Comando: CONO
Punto di base: *immettere punto*
Diametro/<Raggio> della base: *distanza o "D"*
Diametro/<Raggio> dell'apice: *distanza "D" o RETURN*
Altezza: *immettere distanza*
Numero di segmenti <16>: *valore intero o RETURN*



A.3.1.3 CUPOLA o SCODELLA - Rete poligonale emisferica

Questi comandi servono a creare una cupola o una scodella (intesa come parte superiore o inferiore) partendo da un punto centrale e un raggio o diametro. E' possibile anche specificare il numero di segmenti in ogni direzione che verranno disegnati sulla superficie dell'oggetto. (una scodella è costruita come una cupola "capovolta"). La sequenza di comando per creare una cupola è riportata qui sotto.

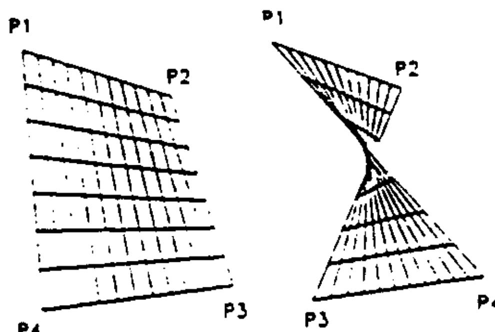
> Comando: <u>CUPOLA</u>
> Centro della cupola: <u>*immettere punto*</u>
> Diametro <Raggio>: <u>*distanza o "D"*</u>
> Numero di segmenti longitudinali <16>: <u>*valore intero o RETURN*</u>
> Numero di segmenti latitudinali <8>: <u>*valore intero o RETURN*</u>



A.3.1.4 RETE - Rete planare semplice

RETE costruisce una rete planare ma, invece di specificare ogni vertice come per il comando 3DRETE, basta specificare solo i 4 vertici estremi e le dimensioni *M* e *N* della rete. Le posizioni dei vertici intermedi vengono calcolate automaticamente.

Comando: <u>RETE</u>
Primo vertice: <u>*immettere punto*</u>
Secondo vertice: <u>*immettere punto*</u>
Terzo vertice: <u>*immettere punto*</u>
Quarto vertice: <u>*immettere punto*</u>
Dim M rete: <u>*valore intero*</u>
Dim N rete: <u>*valore intero*</u>



Le dimensioni M e N della rete informano AutoCAD sul numero di linee da disegnare in ogni direzione lungo la rete; occorre fornire un numero intero tra 2 e 256 in entrambi i casi. AutoCAD evidenzia gli spigoli in questione per far riconoscere le direzioni M e N.

Bisogna selezionare i quattro vertici in senso orario o antiorario, altrimenti la rete risultante si intersecherà con sè stessa e il risultato sarà come nella figura a destra.

A.3.1.5 PIRAMIDE

Questo comando costruisce piramidi a tre o quattro facce. Potete scegliere scegliere se le facce laterali della piramide devono incontrarsi in un unico punto (apice) oppure specificare tre o quattro punti per formare una faccia superiore. Le facce laterali di una piramide a base quadrata possono incontrarsi anche lungo una cresta definita da due punti. Notate che l'altezza dell'apice, della faccia superiore o della cresta della piramide è determinata dal valore della Z dei punti forniti.

Comando: <u>PIRAMIDE</u>
Primo punto base: <u>*immettere punto*</u>
Secondo punto base: <u>*immettere punto*</u>
Terzo punto base: <u>*immettere punto*</u>
Tetraedro/<Quarto punto base>: <u>*punto oppure "T"*</u>

Se rispondente con "Tetraedro" all'ultima richiesta, AutoCAD presenta il seguente messaggio:

Faccia superiore/<Punto apice>: <u>*punto oppure "FS"*</u>

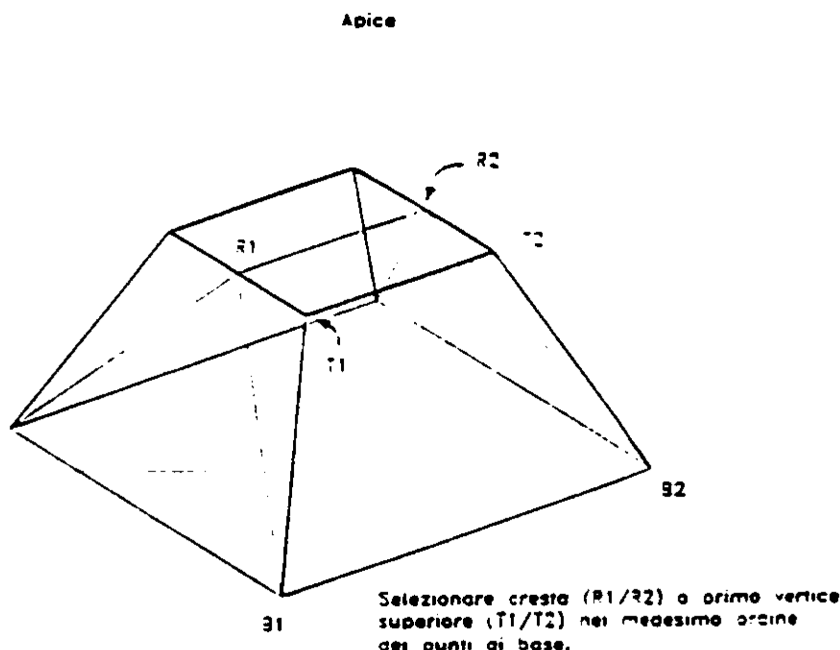
Se rispondete con un punto, AutoCAD genera una piramide a tre facce che si incontrano in un apice costituito da un solo punto. Se rispondente con "Faccia superiore" (o "FS"), AutoCAD chiede i tre punti che specificano il poligono superiore.

Se selezionate l'opzione "Quarto punto base" alla richiesta "Tetraedro/<Quarto punto base>:" viene presentata una serie di opzioni differenti:

Cresta/Faccia Superiore/<Apice>:

Le opzioni Faccia Superiore e Apice sono identiche a quelle relative al tetraedro. L'opzione "Cresta" chiede i due estremi della linea che formerà la cresta della piramide. Tali punti devono essere immessi nella stessa direzione dei punti di base, il primo estremo deve cioè trovarsi sopra al primo punto di base in modo da evitare che la piramide si intersechi con sè stessa. Lo stesso vale per i punti da immettere il risposta all'opzione Faccia Superiore.

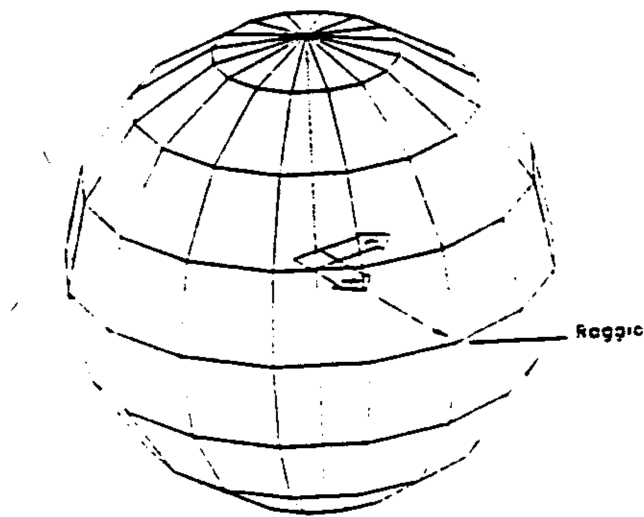
L'illustrazione seguente mostra l'effetto delle opzione "Cresta" e "Faccia Superiore" su una piramide a quattro facce laterali. I punti B1/B2 indicano l'ordine nel quale i punti di base sono stati selezionati (direzione antioraria). I punti R1/R2 e T1/T2 mostrano l'ordine (corrispondente a B1/B2) nel quale devono essere specificati i punti della cresta e della faccia superiore.



A.3.1.6 SFERA

Questo comando costruisce una sfera a rete poligonale. Si può controllare il numero di segmenti generati in ogni direzione sulla superficie della sfera. La sequenza di comando è la seguente:

Comando: SFERA
Centro della sfera: *immettere punto*
Diametro/<Raggio>: *distanza o "D"*
Numero dei segmenti longitudinali <16>: *valore intero o RETURN*
Numero dei segmenti latitudinali <16>: *valore intero o RETURN*

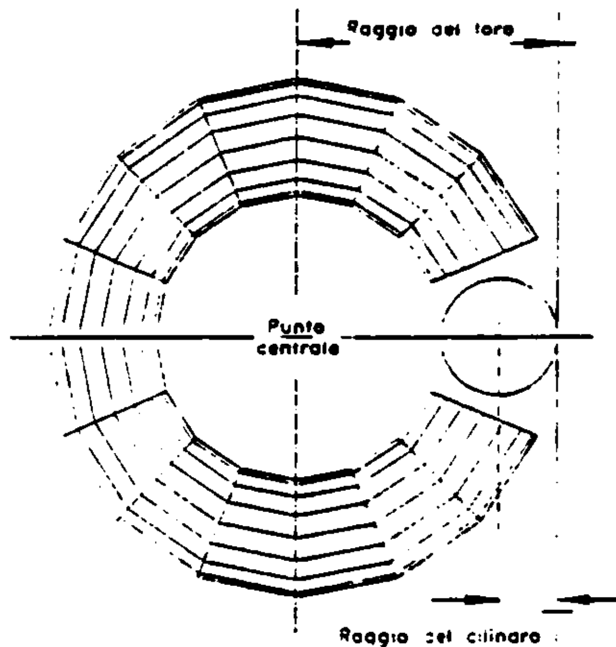


A.3.17 TORO

Questo comando genera una rete poligonale di forma toroidale. Il tubo del toro è costruito ruotando un cerchio intorno ad una linea disegnata sul piano del cerchio e parallela all'asse Z corrente. Il centro del toro giace sul piano di costruzione corrente.

Comando: TORO
Centro del toro: *immettere punto*
Diametro/<Raggio> del toro: *distanza o D*
Diametro/<Raggio> del tubo: *distanza o D*
Segmenti della circonferenza del tubo <16>: *valore intero o RETURN*
Segmenti della circonferenza del toro <16>: *valore intero o RETURN*

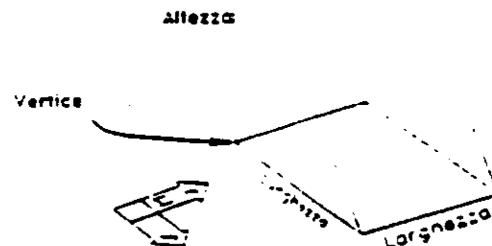
AutoCAD misura il raggio o il diametro del toro dal suo punto centrale al suo spigolo *esterno*. Potete specificare il numero dei segmenti da disegnarsi in ogni direzione.



A.3.1.8 CUNEO

Questo comando definisce una rete poligonale in forma di cuneo con base ad angolo retto. Per prima cosa si comunica ad AutoCAD il punto d'origine del cuneo. Si specifica quindi la lunghezza, larghezza, altezza e angolo di rotazione relativamente a questo punto.

Comando: <u>CUNEO</u>
Vertice del cuneo: *immettere punto*
Lunghezza: *immettere distanza*
Larghezza: *immettere distanza*
Altezza: *immettere distanza*
Angolo di rotazione intorno all'asse Z: *immettere angolo*



L'angolo di rotazione viene specificato relativamente al piano XY corrente; il punto di base della rotazione è il vertice del cuneo.

A.3.2 3DARRAY. Serie 3D rettangolari e polari

Il comando 3DARRAY crea serie rettangolari tridimensionali (se si specifica il numero di righe, di colonne e di livelli) o polari se si specifica un asse di rotazione.

Comando: 3DARRAY
Selezionare oggetti: *mostrare gli oggetti da copiare*
Serie rettangolare o polare (R/P):

Se si sceglie l'opzione Rettangolare (o solo "R"), AutoCAD chiede il numero di righe, colonne e livelli che devono essere costruiti. Il valore standard per ognuna di queste opzioni è <1>.

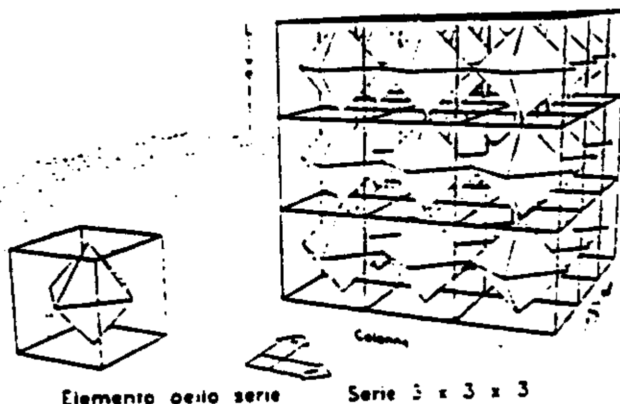
Numero di righe (---)<1>:
Numero di colonne (---)<1>:
Numero di livelli (---)<1>:

Una serie rettangolare viene costruita riproducendo un elemento di base (cioè l'oggetto selezionato) il numero di volte richiesto lungo gli assi X, Y e Z. Di conseguenza una serie in cui sia righe che colonne e livelli hanno valore 1 non ha senso e viene rifiutata dal programma.

AutoCAD richiede poi la distanza tra una riga e l'altra e ripete la stessa richiesta anche per le colonne e i livelli.

Distanza le righe (---):
Distanza le colonne (---):
Distanza i livelli (---):

Dopo che è stato specificato anche questo intervallo, AutoCAD comincia a disegnare la serie. Si può interrompere la costruzione prima che sia terminata immettendo CTRL C.



Per creare una serie polare 3D, selezionate gli oggetti che devono essere riprodotti e rispondete al messaggio che appare quando viene scelta l'opzione "Polare" (o "P"). Il messaggio richiede il numero di copie e l'angolo da riempire:

Numero di copie:
Angolo da riempire <360>:

Digitate il numero di copie che formeranno la serie, compreso l'oggetto originale. Se selezionate più di un oggetto, considerate l'intero gruppo di selezione come un'unità). A differenza del comando SERIE di AutoCAD, non è ammessa una risposta nulla alla domanda "Numero di copie:"

"Angolo da riempire" informa AutoCAD di quanto la serie deve ruotare intorno al centro. Immettete un numero positivo per rotazioni in senso antiorario e un numero negativo per rotazioni in senso orario. Per una serie circolare completa, scegliete il valor standard <360>.

Il prossimo messaggio di richiesta sarà:

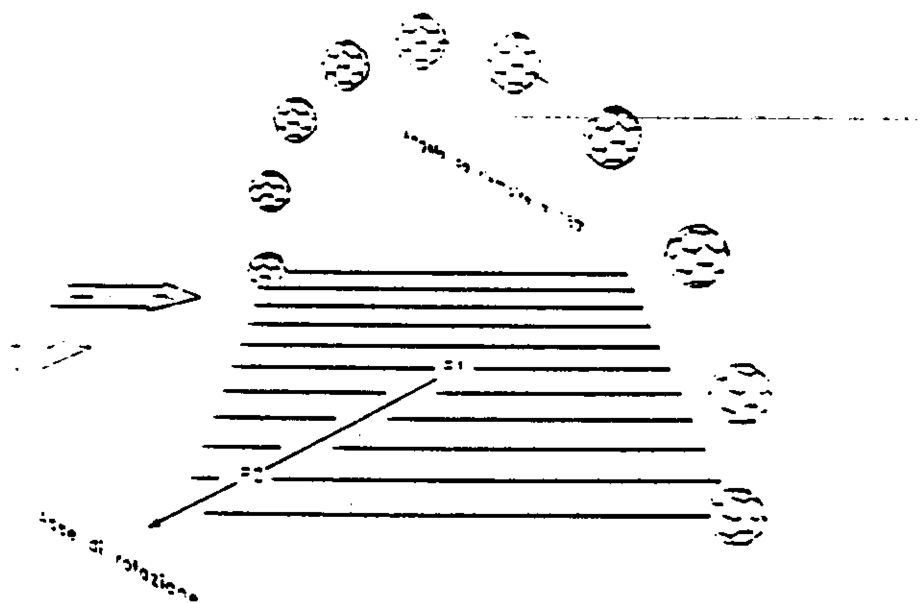
Ruotare gli oggetti mentre vengono copiati? <S>

Una risposta affermativa induce AutoCAD a disegnare ogni copia ruotata in accordo con l'angolo che forma con l'asse di rotazione.

AutoCAD chiede ora di definire l'asse intorno al quale la serie dovrà ruotare. Il comando di AutoCAD SERIE genera una serie polare operando una rotazione intorno all'asse Z corrente. Il comando SERIE3D permette invece di definire un nuovo asse Z temporaneo intorno al quale la serie verrà generata.

Centro della serie:
Secondo punto sull'asse di rotazione:

Nell'illustrazione seguente, il punto centrale (P1) e il secondo punto (P2) definiscono un asse Z temporaneo che giace sul piano XY corrente. La serie risultante è ruotata di 90 gradi relativamente al piano XY.



A.3.3 AFKINET, AFLIX, AFWALK - File di AutoFlix aggiornati

Questi file vengono forniti solo per le versioni di AutoCAD sotto PC-DOS/MS-DOS. Si tratta di versioni aggiornate di file normalmente distribuiti con AutoFlix™. Se siete utenti di AutoFlix, dovreste servirvi di queste versioni con AutoCAD Release 10.

A.3.4 ASCTEXT - Inserimento di un testo proveniente da un file ASCII

Questo comando inserisce file di testo in formato ASCII in disegni di AutoCAD. Potete specificare se il testo da inserirsi deve apparire a sinistra, a destra, centrato o in mezzo. Non viene presupposta nessuna estensione, se il file dispone di un'estensione, questa deve essere specificata esplicitamente. E' possibile anche specificare un prefisso d'elenco come in "/acad/sample.txt" o "\\acad\\sample.txt". La sequenza dei messaggi è la seguente:

> Comando: ASCTEXT
> File da leggersi (compresa estensione): *immettere nome file*
> Punto iniziale o Centro/Mezzo/Destra: *punto o digitare lettere dalla tastiera*
> Altezza <*standard*>: *valore o RETURN*
> Angolo di rotazione <*standard*>: *angolo o RETURN*
> Cambio opzioni di testo? <N>: *"S", "N" o RETURN*

Se rispondete all'ultima richiesta con RETURN, AutoCAD legge all'interno del file da voi specificato (se lo trova). Se rispondete invece con "S", appaiono i messaggi di richiesta seguenti. Potete cambiare i valori per ogni opzione o rispondere con un RETURN per accettare il valore standard presentato.

> Distanza tra le linee(<Auto>):
> Prima linea da leggersi/<1>:
> Numero di linee da leggersi/<Tutte>:
> Sottolineare ogni linea? <N>:
> Sopralineare ogni linea? <N>:
> Tutte le lettere MAiuscole/MInuscole/? <N>:
> Colonne? <N>:

Se rispondete affermativamente all'ultima richiesta, AutoCAD vi chiederà la distanza tra le colonne e il numero di linee per colonna:

> Distanza tra colonne: *immettere distanza*
> Numero di linee per colonna: *valore intero*

Dopo che avete risposto a tutti questi messaggi, AutoCAD scriverà il testo.

A.3.5 ASHADE - File "ashade.lsp" aggiornato

Questo File viene fornito solo per le versioni di AutoCAD sotto PC-DOS/MS-DOS. Si tratta della versione aggiornata del file "ashade.lsp" normalmente distribuito con AutoShade™. Se siete utenti di AutoShade, dovreste usare questo file quando inserite apparecchi fotografici, luci o scene con la release 10 di AutoCAD. I messaggi di richiesta rimangono uguali a quelli riportati nel Manuale all'Uso di AutoShade.

La nuova versione di ASHADE comprende un nuovo comando, APPVISTA. Questo comando può essere utilizzato con la release 10 di AutoCAD (con FLATLAND uguale a 0) per visualizzare il disegno visto in prospettiva da un apparecchio fotografico di AutoShade. APPVISTA chiede di selezionare l'apparecchio desiderato, basta puntare sull'apparecchio e dopo pochi istanti appare sullo schermo il disegno come se fosse visto dall'obiettivo dell'apparecchio. Notate che APPVISTA genera una vista prospettica.

A.3.6 ATTREDEF - Aggiornamento e ridefinizione di attributi

Questo comando permette di ridefinire un blocco e aggiornare gli attributi associati a precedenti inserimenti di tali blocco. Gli eventuali nuovi attributi vengono aggiunti ai parametri del blocco e ricevono i loro valori standard. Attributi preesistenti compresi nella nuova definizione di blocco mantengono i loro valori. Attributi precedenti invece, che non sono più compresi nella nuova definizione vengono eliminati.

> Comando: ATTREDEF
> Nome del blocco da ridefinire: *vecchio nome del blocco*
> Selezionare nuovo blocco ...
> Selezione oggetti: *scegliere gli oggetti che costituiranno il nuovo blocco*
> Punto di inserimento del nuovo blocco: *immettere punto*

Se gli identificatori di entità sono attivi, nuovi identificatori vengono assegnati ad ogni parametro di blocco ridefinito.

A.3.7 CHGTEXT - Ricerca e sostituzione di un testo

Il comando CHGTEXT permette una elementare editazione di entità di testo. Offre la possibilità di specificare una nuova stringa che può sostituire una vecchia stringa trovata nell'ambito di entità di testo selezionate. Gli spazi sono significativi in entrambe le stringhe.

> Comando: CHGTEXT
> Selezioanre oggetti: *selezionare il testo da considerare*
> Vecchia stringa: *immettere testo da sostituire*
> Nuova stringa: *immettere nuovo testo*

A.3.8 DELLAYER - Cancellazione di tutte le entità su un piano

Questo comando permette di cancellare tutte le entità che si trovano su un piano specificato.

> Comando: DELLAYER
> Piano da cancellare: *immettere nome del piano*

DELLAYER si serve del meccanismo di filtro (ssget "X") di AutoLISP per creare un gruppo di selezione comprendente tutte le entità sul piano specificato. Esegue poi il comando ELIMINA considerando il gruppo di selezione come l'entità da eliminare. Questo comando può essere facilmente modificato per permettere l'eliminazione di altri gruppi di entità: ad esempio, è possibile implementare comandi per operare su tutte le entità di un determinato colore, un determinato tipo (ad es. cerchi) o tutte le entità di una certa elevazione. (Rimandiamo al comando SSX descritto in seguito).

A.3.9 EDGE - Modifica la visibilità di spigoli di facce 3D

Questo comando permette di modificare interattivamente la visibilità degli spigoli delle facce tridimensionali.

> Comando: EDGE
> Visualizza/<Selezione lato>:

Il messaggio "Visualizza/<Selezioe lato>:" viene ripetuto finchè non viene fornita una risposta nulla, ciò permette di modificare la visibilità di più di uno spigolo con un solo comando. Selezionando uno spigolo, questo diventa automaticamente invisibile (se è invece già invisibile, torna ad essere visibile). Se gli spigoli di una o più facce si sovrappongono, la visibilità di ogni spigolo sovrapposto ne risulta alterata.

E' possibile modificare solo le caratteristiche di spigoli visibili. L'opzione "Visualizza" evidenzia gli spigoli invisibili di una o più facce 3D. Infatti potete procedere all'editazione di uno spigolo solo se riuscite a vederlo. Se volete rendere visibili tutti gli spigoli invisibili prima di procedere al comando EDGE, potete assegnare alla variabile di sistema SPLFRAME valore 1.

A.3.10 LEXPLODE - Nuova versione del comando ESPLOSO

Il comando di AutoCAD ESPLOSO posiziona le entità di tratteggio e quotatura sul piano 0. Questa versione alternativa del comando ESPLOSO colloca tali entità sul medesimo piano dell'entità originale.

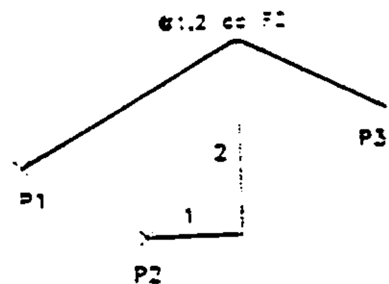
> Comando: <u>LEXPLODE</u>
> Selezionare blocco di riferimento, polilinea, quota o rete: *eseguire*

AutoCAD riporta il nome del piano sul quale sono state collocate le nuove entità.

A.3.11 REF - Ottenimento di un punto relativo

Si tratta di una routine molto utile per ricavare un punto relativo e può essere richiamata ogni volta che un comando di AutoCAD richiede un punto. Basta digitare REF in risposta ad un messaggio di richiesta riguardante un punto e immettere il punto di base (di Riferimento) desiderato e lo spostamento relativo o polare rispetto questa base. Ad esempio:

> Comando: <u>LINEA</u>
> Dal punto: *punto P1*
> Al punto: <u>(ref)</u>
> Punto di riferimento: *punto P2*
> Immettere coordinata relativa/polare (con @): *@1,2*
> Al punto: *punto P3*
> Al punto: <u>RETURN</u>



Come vedete nella figura, questa sequenza crea una coppia di linee che si incontrano in un punto spostato di 1 unità X e 2 unità Y dal punto di riferimento (P2).

A.3.12 SETUP - Definizione della scala e dei limiti di un disegno

Solitamente si richiama SETUP (in italiano IMPOST) dal menù di schermo di AutoCAD. Questo comando imposta le unità di disegno e i limiti di un nuovo disegno sulla base delle dimensioni del foglio e del fattore scalare da voi scelto. La routine termina disegnando un bordo formato da un linea semplice per permettere un migliore orientamento all'interno dei limiti del disegno.

A.3.13 SSX - Versione semplificata di (ssget "x")

Questa routine fornisce una maniera pratica di selezione entità sulla base del tipo, del colore, tipo di linea, piano, nome del blocco, stile di testo e altezza. Ad esempio potete usare SSX per selezionare tutte le linee di colore verde. Potete richiamare SSX in risposta al messaggio di richiesta "Comando:", come illustrato sotto, oppure in risposta ad ogni messaggio che richiede la selezione di oggetti.

Comando: (ssx)
\>> Nome del blocco/Colore/Entita/Piano/Tipolinea/Stile/Altezza:

Se si seleziona un'opzione, AutoCAD richiede il valore da immettere. Il messaggio di richiesta principale viene ripetuto finchè non si fornisce una risposta nulla; AutoCAD quindi esamina il disegno alla ricerca delle entità che rispondono ai criteri definiti. (E' ovvio quindi che si puo specificare un solo valore per ogni opzione).

Se richiamate SSX in risposta a "Comando:", potete utilizzare il nuovo gruppo di selezione immettendo "P" (o "Precedente") al prossimo messaggio di selezione oggetti. Se rispondete con SSX al messaggio di selezione oggetti il nuovo gruppo di selezione viene passato direttamente al comando in attesa. Nell'esempio seguente SSX e stato richiamato durante il comando SPOSTA.

Comando: SPOSTA
Selezionare oggetti: (ssx)
\>> Blocco/Colore/Entita/Piano/Tipolinea/Stile/Altezza: E
\>>Tipo di entità: Linea
\>> Blocco/Colore/Entità/Piano/Tipolinea/Stile/Altezza: C
\>> Numero colore: 1
\>>Blocco/Colore/Entità/Piano/Tipolinea/Stile/Altezza: RETURN
n trovato
Selezionare oggetti: RETURN
Punto di base o spostamento: ...

Entità possono disporre di un proprio colore e tipo di linea oppure possono ereditare queste caratteristiche dal piano sul quale giacciono (colore e tipolinea "DA PIANO"). SSx indivua entità con un colore e un tipo di linea specificati solo se queste caratteristiche sono state assegnate all'entità individualmente.

A.4 Esempi di programmazione

I programmi di AutoLISP contenuti nel dischetto "Bonus" hanno lo scopo di dimostrare le possibilità di AutoLISP e di illustrare i metodi di cui servirsi per scrivere i programmi in AutoLISP. Benchè la maggior parte di questi programmi eseguano compiti piuttosto utili, non sono provvisti di funzioni per la verifica dell'immissione esatta o di situazioni di errore e le routine non sono così perfette come negli esempi precedenti. Si invitiamo quindi a esaminare il file sorgente di questi programmi per modificarli o estenderli, se lo desiderate. I capitoletti seguenti descrivono brevemente questi programmi modello.

A.4.1 AXROT - Rotazione di entità intorno ad un asse

Questa routine ruota una o più entità di un numero di gradi specificato intorno agli assi X, Y e Z. Potete realizzare una serie di rotazioni premendo RETURN per ripetere il comando ROTAS e immettere "P" (per "Precedente") al messaggio di richiesta "Selezione oggetti".

Comando: AXROT
Selezionare oggetti: *eseguire*
Assi di rotazione X/Y/Z: *"X", "Y" o "Z"*
Angolo di rotazione <0>: *valore o RETURN*
Punto base <0, 0, 0>: *punto o RETURN*

A.4.2 CHFACE - Spostamento dei vertici di una faccia 3D

Questa routine fornisce un metodo per spostare i vertici di una faccia 3D. E' particolarmente utile nei casi in cui due vertici si sovrappongono (come in una faccia 3D di tipo triangolare) e desiderate spostarne uno. Potete selezionare il vertice desiderato tramite puntamento o tramite l'ordine (primo, secondo, ecc.) in cui i vertici sono stati immessi.

Comando: CAMFAC
Selezionare entità da cambiare: *eseguire*
1/2/3/4/Annulla/Visualizzazione/<Selezionare vertici>:

A.4.3 CL - Costruzione di linee centrali

Questa routine costruisce una coppia di linee centrali passanti per il centro di un cerchio. CL disegna queste linee in modo che siano parallele all'UCS corrente al momento della costruzione del cerchio e le pone su un piano "CL" (creando questo piano se già non esiste).

Comando: CL
Selezionare arco o cerchio: *eseguire*
Raggio è nnn
Lunghezza/<Estensione>: *distanza o "L"*

Se immettete "Lunghezza" (o semplicemente "L"), AutoCAD chiede di specificare la lunghezza di *un braccio* della linea centrale; una linea elastica viene attaccata al centro del cerchio e la riga monitor di AutoCAD fornisce un conto progressivo della distanza. Altrimenti potete immettere il valore della distanza alla quale le linee centrali si estendono al di fuori della circonferenza del cerchio.

A.4.4 DRAWMAN - Esempio di gestione di entità

Se utilizzato con il disegno modello REVINFO, questo programma implementa un sistema rudimentale di controllo e revisione del disegno. E' stato fornito per scopi puramente illustrativi, per mostrarvi ciò che potete fare servendovi dei gestori di entità e di una piccola porzione di AutoLISP. Questo programmino non ha la pretesa di essere nè un gestore di disegni multifunzionale nè la base per l'implementazione di un tale gestore.

Dopo aver caricato il programma, potrete scegliere tra i comandi seguenti:

LOGON Domanda un chiave d'accesso, cioe il vostro nome, un numero ECO (Engineering change order) e commenti di vario tipo. Se già era presente un'altra chiave, questa viene automaticamente annullata.

LOGOUT Annulla la vostra chiave d'accesso.

END Ridefinisce il comando FINE di AutoCAD in modo tale che venga annullata la chiave d'accesso di utenti attivi, nel caso ne esistessero.

RLIST Presenta una lista di tutte le revisioni apportate al disegno.

FINGER Richiede di puntare su un'entità. Informa su chi ha creato l'entità, quando, quale ECO è stato aggiunto ed eventuali commenti relativi alla revisione in questione.

SELUSER Richiede il nome di un utente. Evidenzia tutte le entità del disegno che sono state aggiunte da questo utente. Premendo un tasto di funzione, tali entità vengono posizionate all'interno di un gruppo di selezione "Precedente" in modo che possiate operare su di esse con altri comandi di AutoCAD.

SELECO Richiede un numero ECO. Evidenzia tutte le entità nel disegno relative a tale ECO. Premendo un tasto di funzione, tali entità vengono posizionate all'interno di un gruppo di selezione "Precedente" in modo che possiate operare su di esse con altri comandi di AutoCAD.

Le informazioni di controllo sulle revisioni sono memorizzate nel disegno sotto forma di blocchi con attributi invisibili sul piano "SREV".

A.4.5 FACT - Calcolo fattoriale

Questa routine illustra l'uso della ricursione per calcolare il fattoriale di un intero.

Comando: FACT
Immettere un intero: *eseguire*

A.4.6 FCOPY - Copia di un file di testo in un'altro

Questo programma prende come argomenti il nome di due file di testo ASCII e copia il primo file nel secondo. Se il primo file non esiste, viene visualizzato un messaggio di errore; se e invece il secondo file che non esiste, viene creato. Se il secondo file è già esistente, quanto precedentemente conteneva viene sovrascritto.

Comando: (fcopy *"filepartenza.ext" "filedestinazione.ext"*)

Questo programma è da utilizzarsi solo con file di testo ASCII. Non usatelo mai con file binari.

A.4.7 FPLOT - Funzione di riproduzione grafica di due variabili

Questa routine genera una rete poligonale tridimensionale rappresentante i valori di una funzione a due variabili nell'ambito di un intervallo specificato di valori per le due variabili, con una risoluzione definita (specificata dal numero di suddivisioni all'interno dell'intervallo). quando richiamate FPLOT, dovete fornire quattro argomenti:

Comando: (fplot *<funzione> <intervallox> <intervalloy> <risoluzione>*)

<funzione> è la funzione che deve essere valutata. Solitamente si tratterà del nome tra virgolette di una funzione definita precedentemente o una definizione LAMBDA di una funzione.

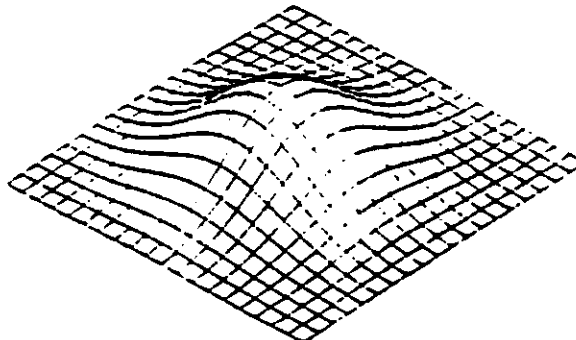
<intervallox> e *<intervalloy>* sono liste che specificano l'intervallo per i valori X e Y rispettivamente. Il primo elemento di ogni lista è il limite inferiore, il secondo elemento il limite superiore.

<risoluzione> è un numero intero che specifica la risoluzione della rete approssimante la superficie definita dai valori della funzione che fungono da argomenti nell'intervallo specificato.

Ad esempio, per riprodurre graficamente (e**(-(X**2 + Y**2))) in un intervallo da -2 a 2 per entrambi gli assi (con 20 tabulazioni lungo ogni asse), digiterete:

```
(fplot '(lambda (x y) (exp (- (+ (* x x) (* y y)))))
       '(-2 2)
       '(-2 2)
        20
)
```

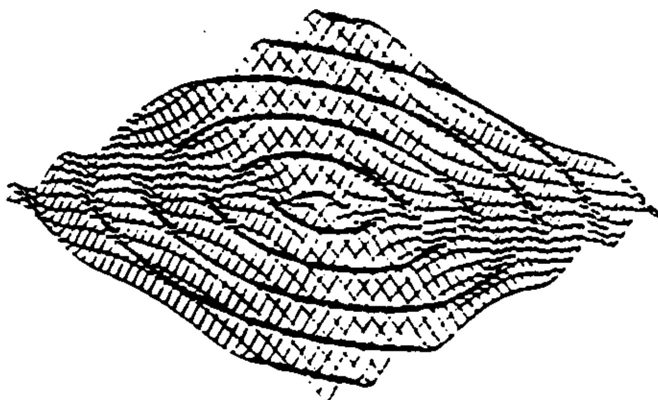
Il risultato visualizzato con punto di vista pari a 1,-1,1 sarà una palla da tennis nascosta sotto un tappeto.



Potete riprodurre graficamente anche una funzione predefinita. Ad esempio:

```
(defun cs (x y)
     (cos (sqrt (+ (* x x 2) (* y y))))
)
(visualf 'cs '(-20 20) '(-20 20) 40)
```

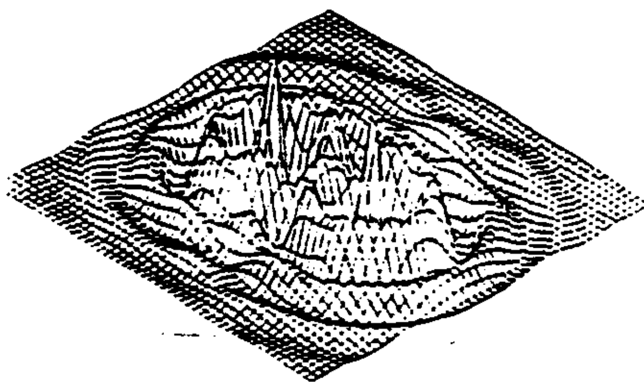
Il risultato sarà una serie di onde ellittiche, come quelle che si generano uno stagno nel quale avete appena gettato un sasso.



Il file "fplot.lsp" contiene anche un caso predefinito più complesso. Se immettete:

Comando: DEMO

otterrete il modello d'interferenza di due onde cosinusoidali smorzate esponenzialmente. Questo esempio vi illustra la pluralità di superfici complesse che potete generare con una semplice definizione servendovi di FPLOT.



A.4.8 FPRINT - Lista file di testo sullo schermo

Questa funzione presenta sullo schermo una lista di file di testo ASCII. Il passaggio allo schermo di testo non è automatico.

Comando: (fprint "*nomefile.ext*")

A.4.9 PROJECT - Proiezi 3D sull'UCS corrente

Questa routine implement iezione "piana" di modelli 3D a reticolo (linee, archi. cerchi, polilinee, poligoni) sull'UCS corrente. Può trattarsi di un utile sussidio per generare disegni di b D. Reti tridimensionali non possono essere proiettate. Informazioni relative all polilinee) e l'estrusione non vengono prese in considerazione.

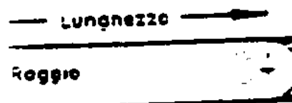
A.4.10 RPOLY - Modific 2

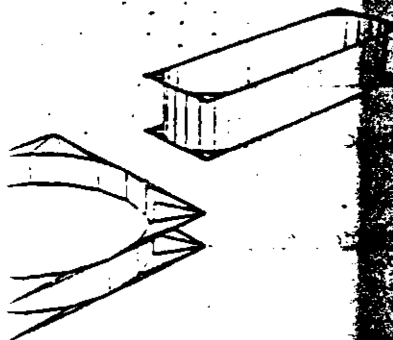
Questa routine ridefinisc asuale sostituendo iterativamente i suoi vertici con i punti mediani dei suoi s difica trasforma la maggior parte dei poligoni casuali in poligoni convessi di fo

A.4.11 SLOT - Costruzi e cavità

Questa routine si serve per costruire "cavità" e "scanalature" per modelli da rappresentare con l'aiut Viene generato uno spigolo rettangolare con facce invisibili intorno agli est atura o della cavità. La sequenza per la costruzione di una scanalatura è illustr uito. Nell'illustrazione, SPLFRAME ha valore 0 per permette di vedere lo sp

Comando: <u>SLOT</u> <u>TURN</u>
CAvità o scalatu
Primo centro de *ettere punto*
Raggio della sc *e distanza*
Secondo centro *immettere punto*
Profondità: *imm*





A.4.12 SPIRAL - Costruzione di spirali 2D

Costruisce una spirale bidimensionale dato un punto di base, la crescita per rotazione e i punti per cerchio.

Comando: SPIRAL
Punto centrale: *immettere punto*
Numero di rotazioni: *immettere numero*
Passo per rotazione: *immettere distanza*
Punti per rotazione: *immettere numero*

A.4.13 SQR - Calcolo della radice quadrata

Implementa una funzione di radice quadrata in AutoLISP utilizzando il medesimo metodo di AutoCAD.

Comando: (sqr *<numero>*)

A.4.14 TABLES - Visualizzazione/riordino di tavole di simboli

Questa routine contiene un numero di funzioni che applicano le funzioni TBLNEXT e TBLSEARCH di AutoLISP. Le funzioni (piano), (tipolinea), (vista), (blocco), (stile), (ucs) e (finestra) possono essere chiamate indipendentemente. Ognuna elenca le entrate nella tavola di simboli associata, a richiesta in ordine alfabetico. Il comando TABLES chiama queste funzioni una dopo l'altra.

Per le tavole relative ai piani, ai tipi di linea e agli stili di testo, un asterisco nella colonna uno indica l'impostazione corrente. Se il tipo di linea corrente è "DAPIANO", il tipo di linea corrispondente al piano corrente verrà segnata con una "L" nella colonna uno.

Comando: TABLES o *(nome della tavola)*

Appendice B

Messaggi di errore

Quando AutoLISP individua una condizione di errore, cancella la funzione corrente e richiama la funzione *ERROR* definita dall'utente con un messaggio indicante il tipo di errore. Se l'utente non ha definito nessuna funzione *ERROR* (*ERROR* è legato a NIL), verrà visualizzato il formato generale del messaggio d'errore cioè:

error: *messaggio*

seguito dalla traccia della funzione. Se esiste una funzione *ERROR* definita dall'utente questa viene richiamata con *messaggio* quale unico argomento.

B.1 Errori nei programmi utente

Presentiamo una lista di messaggi di errore che possono apparire durante la creazione e la messa a punto di funzioni di AutoLISP. La maggior parte di questi messaggi messaggi indica tipici errori di programmazione in LISP come:

- nomi di funzione o di simboli scritti non correttamente
- erroneo numero o tipo di argomenti di funzione
- errori di parentesi
- errori di virgolette (strighe non terminate)
- l'esecuzione di una funzione non è ancora terminata e già si cerca di utilizzarne i risultati

Benchè questi messaggi indichino solitamente errori di programmazione da parte dell'utente, possono anche essere il risultato di imprecisioni nella programmazione originale di AutoLISP stesso. Se non riuscite ad identificare il possibile errore da voi commesso, vi preghiamo di riempire un foglio di descrizione del possibile errore interno ad AutoLISP e di spedirlo alla Autodesk AG.

Atomlist modified after function swap (atomlist modificato durante paginazione attiva)

Un'espressione ha modificato ATOMLIST. Ciò non è permesso mentre la funzione di paginazione VMON è attivata.

AutoCAD rejected function (funzione rifiutata da AutoCAD)

Gli argomenti forniti a una funzione di AutoCAD non sono validi (esempio: SETVAR di una variabile di sistema "read-only" o TBLNEXT con un nome di tabelle non valido) o la funzione stessa non è valida nel contesto corrente. Ad esempio, non si può usare una funzione GETxxx all'interno della funzione COMMAND.

bad argument type (tipo di argomento scorretto)

A una funzione è stato passato un argomento scorretto. (Ad esempio, non si può chiamare STRLEN con un intero.)

bad association list (lista delle associazioni scorretta)

La lista fornita a ASSOC non consiste di liste "(chiave valore)".

bad entmod list (lista di ENTMOD scorretta)

L'argomento passato a ENTMOD non è una lista appropriata di dati di entità (come quelle restituite da ENTGET).

bad entmod list value (valore scorretto di lista ENTMOD)

Una delle sottoliste nella lista delle associazioni fornita ad ENTMOD contiene un valore scorretto.

bad formal argument list (lista di argomenti formali scorretta)

Durante la valutazione di questa funzione, AutoLISP ha trovato una lista di argomenti formali non valida. Forse la funzione non è una funzione vera e propria, ma piuttosto una lista di dati.

bad function (funzione scorretta)

Il primo elemento della lista non è un nome di funzione corretto. Forse è il nome di una variabile o un numero. Questo messaggio può anche indicare che la funzione digitata non è stata definita correttamente; per es., se è stata omessa la lista degli argomenti formali richiesta.

bad list (lista scorretta)

A una funzione è stata passata una lista formata in modo scorretto. Ciò può succedere se un numero reale inizia con un punto decimale; in questo caso bisogna digitare uno zero davanti al punto.

bad node (nodo scorretto)

Tipo di elemento non valido incontrato dalla funzione TYPE.

bad node type in list (tipo di nodo scorretto nella lista)

Tipo di elemento non valido incontrato dalla funzione FOREACH.

bad point argument (argomento di punto scorretto)
bad point value (valore di punto scorretto)

Un punto (lista di due numeri reali) definito in modo scorretto è stato fornito a una funzione che richiedeva un punto. Non è permesso iniziare un numero reale con un punto decimale; in questo caso bisogna digitare uno zero davanti al punto.

bad ssget list (lista SSGET scorretta)

L'argomento passato a (SSGET "X") non è una lista di dati di entità appropriata (come restituito da ENTGET).

bad ssget list value (valore di lista SSGET scorretto)

Una delle sottoliste nella lista delle associazioni fornita a (SSGET "X") contiene un valore scorretto.

base point is required (richiesto punto di base)

La funzione GETCORNER è stata richiamata senza fornire l'argomento del punto di base.

boole arg1 <0 or >15

Il primo argomento della funzione BOOLE deve essere un intero compreso fra 0 e 15.

can't evaluate expression (impossibile valutare l'espressione)

Questo errore puo essere causato dalla collocazione scorretta di un punto decimale o da altre espressioni formate in modo scorretto.

Can't open (file) for input -- LOAD failed (impossibile aprire il file di input -- LOAD non eseguibile)

Il file fornito alla funzione LOAD non è reperibile o l'utente non ha accesso di lettura al file.

Can't reenter AutoLISP (impossibile ritornare in AutoLISP)

Il buffer di comunicazione AutoCAD - AutoLISP è occupato da una funzione attiva, nessuna nuova funzione puo essere richiamata finchè la funzione non è completa.

console break (*Annullato*)

Una funzione in corso è stata interrotta mediante CTRL C.

divide by zero (divisione per zero)

La divisione per zero non è permessa.

divide overflow (superamento dei limiti a causa di una divisione)

La divisione per un numero molto piccolo ha prodotto un quoziente non valido.

extra right paren (parentesi di chiusura supplementare)

Sono state trovate una o più parentesi di chiusura supplementari.

file not open (il file non è aperto)

Il descrittore di file per operazioni I/O non è quello di un file aperto.

File read - insufficient disk space (lettura file - spazio su disco insufficente)

Lo spazio di stringa è stato esaurito mentre si leggeva da un file. Vedi Capitolo 6.

File size limit exceeded (file troppo grande)

(solo per sistemi sotto UNIX) Un file ha superato i limiti di grandezza posti dal sistema.

Floating-point exception (errore di virgola mobile)

(solo per sistemi sotto UNIX) Il sistema operativo ha scoperto un errore nell'aritmetica a virgola mobile.

function cancelled (funzione annullata)

L'utente ha digitato CTRL C in risposta a un messaggio che sollecitava immissione di dati.

function undefined for argument (argomento non è uno di quelli previsti dalla funzione)

L'argomento passato a LOG o SQRT non rientra nella gamma dei valori permessi.

function undefined for real (funzione non definita per numeri reali)

A una funzione che richiede un numero intero è stato fornito un numero reale. Esempio: (LSH val 1.2)

improper argument (argomento scorretto)

L'argomento di GCD è negativo o uguale a 0.

inappropriate object in function (funzione inadeguata)

Una funzione costruita in maniera inappropriata è stata rilevata dal paginatore della funzione VMON.

incorrent number of argument (numero di argomenti scorretto)

Una funzione QUOTE esige solo un argomento, è stato invece fornito un numero superiore di argomenti.

incorrect number of arguments to a function (numero scorretto di argomenti di una funzione)

Il numero degli argomenti per la funzione definita dall'utente non corrisponde al numero di argomenti formali specificati da DEFUN.

incorrect request for command list data (richiesta scorretta di comando di liste dati)

E' stata incontrata una funzione COMMAND, ma non può essere eseguita essendo attiva un'altra funzione.

input aborted (input annullato)

E' stato rilevato un errore o una condizione di fine file prematura che ha causato l'interruzione dell'input dal file.

insufficient node space (memoria nodale insufficiente)

La memoria nodale "heap" non è sufficiente per eseguire l'azione richiesta. Vedi Cap. 6.

insufficient string space (spazio di memoria per stringhe insufficiente)

La memoria di "heap" non è sufficiente per gestire la stringa di testo specificata. Vedi Cap. 6.

invalid argument (argomento non valido)

Tipo di argomento scorretto o argomento fuori dalla gamma dei valori permessi.

invalid character (carattere non valido)

L'espressione contiene un carattere non valido.

invalid dotted pair (coppia puntata non valida)

Le coppie puntate sono liste speciali contenenti due elementi separati da una costruzione "spazio-punto-spazio". Questo errore puo verificarsi quando si inizia un numero reale con un punto decimale: in questo caso bisogna digitare uno zero davanti al punto.

invalid integer value (valore intero scorretto)

E' stato incontrato un numero minore dell'intero più piccolo permesso o maggiore dell'intero più grande permesso.

LISPSTACK overflow (superamento dei limiti di memoria)

AutoLISP ha debordato al di fuori dello spazio di memoria ausiliaria. Ciò può essere dovuto ad un eccessivo ricorso a funzioni o a liste di argomenti di funzioni particolamente lunghe. Provate ad aumentare il valore della variabile d'ambiente LISPSTACK.

malformed list (lista costruita in modo erroneo)

Una lista letta da un file è terminata prematuramente.

malformed string (lista costruita in modo erroneo)

Una stringa letta da un file è terminata prematuramente.

misplaced dot (punto posizionato erroneamente)

Può verificarsi quando un numero reale inizia con un punto decimale: in questo caso bisogna digitare uno zero davanti al punto.

null function (funzione nulla)

Vi è stato un tentativo di valutare un funzione che ha definizione NIL.

quit / exit abort (interruzione dovuta a quit/exit)

Questo è il risultato di un richiamo delle funzioni QUIT o EXIT, che attualmente non sono in uso in AutoLISP.

too few arguments (non ci sono abbastanza argomenti)

Ad una funzione predefinita non sono stati forniti abbastanza argomenti.

too many arguments (ci sono troppi argomenti)

Ad una funzione predefinita sono stati forniti troppi argomenti.

B.2 Errori interni

I seguenti messaggi di errore dovrebbero essere estremamente rari. Si tratta di messaggi che si riferiscono ad errori interni al programma AutoLISP stesso e dovrebbero essere comunicati alla Autodesk AG se si presentano.

Bad argument to system call

(Solo per sistemi sotto UNIX) Il sistema operativo ha individuato una chiamata di sistema scorretta generata da AutoLISP.

Bus error

(Solo per sistemi sotto UNIX) Il sistema operativo ha individuato un errore di bus.

COMREG re-use overflow
COMREG swapin failure
COMREG swapout failure

Si è verificato un errore di comunicazione AutoCAD-AutoLISP durante la funzione di paginazione VMON.

could not open page file

Il file di paginazione della funzione VMON non può essere aperto.

Double lock
Double unlock

Si è verificato un errore di comunicazione AutoCAD-AutoLISP durante la funzione di paginazione VMON.

error swapping in function
error swapping out function

Si è verificato un errore di I/O durante la funzione di paginazione VMON.

Hangup

(Solo per sistemi sotto UNIX) Il sistema operativo ha individuato un segnale di sospensione.

Illegal instruction

(Solo per sistemi sotto UNIX) Il sistema operativo ha individuato un'istruzione macchina scorretta.

Segmentation violation

(Solo per sistemi sotto UNIX) Il sistema operativo ha individuato un tentativo di indirizzo al di fuori dell'area di memoria di elaborazione

unexpected signal nnn

(Solo per sistemi sotto UNIX) Il sistema operativo ha ricevuto un segnale inatteso.

Indice Analitico

' (funzione QUOTE) 62
* (funzione) 26
+ (funzione) 25
- (funzione) 25
/ (funzione) 26
/= (funzione) 27
1+ (funzione) 28
1- (funzione) 28
< (funzione) 27
<= (funzione) 27
= (funzione) 26
> (funzione) 27
>= (funzione) 28
~ (funzione) 28
3D.lsp 104
3DARRAY comando 112

A

ABS (funzione) 28
Acad.lsp (file) 9, 54
ACADXMEM variabile d'ambiente 97
Accesso alle entità 87
Aggiungere comandi ad AutoCAD 39
ALLOC (funzione) 100
Allocazione manuale 100
AND (funzione) 29
ANGLE (funzione) 29
ANGTOS (funzione) 29
Apostrofo 5
APPEND (funzione) 30
APPLY (funzione) 30
Archiviazione di simboli 99
Aritmetiche, funzioni
- * 26
- + 25
- - 25
- ABS 28
- EXP 42
- EXPT 42
- GCD 44
- LOG 54
- LSH 55
- MAX 56
- MIN 57
- REM 65
- SQRT 68

ASCII (funzione) 30
ASCTEXT comando 114
Ashade.lsp 114
ASSOC (funzione) 31
ATAN (funzione) 31
ATOF (funzione) 31
ATOI (funzione) 32
ATOM (funzione) 32
ATOMLIST 97
ATTREDEF comando 115
AutoCAD, funzioni di
- COMMAND 35
- GRAPHSCR 48
- OSNAP 60
- SETVAR 67
- TEXTSCR 69

AutoFlix files aggiornati 113
AXROT routine 118

B

Base, funzioni di
- APPLY 30
- ASSOC 31
- ATOF 31
- ATOM 32
- BOUNDP 33
- CADR 33
- CAR 34
- CDR 34
- COND 37
- CONS 37
- DEFUN 38
- EVAL 42
- FOREACH 43
- IF 49
- LAST 52
- LENGHT 53
- LIST 53
- LISTP 53
- LOAD 39
- MAPCAR 55
- MEMBER 56
- MINUSP 57
- NOT 57
- NTH 58
- NULL 58
- NUMBERP 58
- PROGN 62
- QUOTE 5, 62
- REPEAT 65
- REVERSE 65
- SET 66
- SETQ 66
- SUBST 31, 68
- TYPE 72
- WHILE 74
- ZEROP 75

Biblioteca di funzioni 9
Blocchi (accesso alla tavola dei simboli) 88
BOOLE (funzione) 32

BOUNDP (funzione) 33

C

CXXXX, funzione 20
CADR (funzione) 33
Cambio schermo 48, 69
CAR (funzione) 34
Caricamento automatico 9, 39, 53
Caricamento dei programmi AutoLISP 102
CDR (funzione) 34
CHFACE routine 118
CHGTEXT comando 115
CHR (funzione) 35
CL routine 118
CLOSE (funzione) 35
CMDECHO variabile di sistema 35
Comandi, ripetizione sullo schermo 35
COMMAND (funzione) 35
COND (funzione) 37
CONO comando 106
CONS (funzione) 37
Convenzioni di AutoLISP 4
Convenzioni relative alla trascrizione 5
Conversione, funzioni di
  ANGTOS 29
  ATOI 32
  FIX 43
  FLOAT 43
  ITOA 52
  RTOS 65
COS (funzione) 38
Costruzione di oggetti tridimensionali 104
CUNEO comando 111
CUPOLA comando 107
Curve a "V" 88

D

Dati di entità, funzioni 83
DCS sistema di coordinate di AutoCAD 70
DEFUN (funzione) 9, 38
DELLAYER comando 115
DIMZP variabile di quotatura 29
Dispositivi, funzioni di accesso
  GRCLEAR 91
  GRDRAW 91
  GRREAD 92
  GRTEXT 91
DISTANCE (funzione) 40
DRAWMAN routine 118

E

Eco dei comandi (CMDECHO) 35
ECS Sistema di coordinate di AutoCAD 85
ECS sistema di coordinate di AutoCAD 70
EDGE comando 115
EDITPL comando di AutoCAD 88
ENTDEL (funzione) 83, 87
ENTGET (funzione) 83
Entità, funzioni di accesso
  ENTDEL 83
  ENTGET 83
  ENTLAST 81
  ENTMOD 86
  ENTNEXT 81
  ENTSEL 82
  ENTUPD 87
  HANDENT 82
  REDRAW 64
  SSADD 80
  SSDEL 80
  SSGET 77
  SSLENGHT 79
  SSMEMB 81
  SSNAME 79
ENTLAST (funzione) 81
ENTMOD (funzione) 86, 87
ENTNEXT (funzione) 81
ENTSEL (funzione) 82
ENTUPD (funzione) 87
EQ (funzione) 41
EQUAL (funzione) 41
ERROR (funzione) 75
Errore, messaggi 125
Errori, gestione 6, 75
EVAL (funzione) 42
EXP (funzione) 42
EXPAND (funzione) 100
EXPT (funzione) 42
EXTLISP programma 97

F

FACT routine 119
FCOPY routine 119
FINDFILE (funzione) 42
FIX (funzione) 43
FLATLAND variabile di sistema 25, 41, 45, 47, 49, 51, 60, 83, 90
FLOAT (funzione) 43
FOREACH (funzione) 43
Formato di release 7
FPLOT routine 120
FPRINT routine 121

Funzioni di AutoLISP influenzate da
FLATLAND 25

G

GCD (funzione) 44
Geometriche, funzioni
ANGLE 29
DISTANCE 40
INTERS 51
POLAR 60
GETANGLE (funzione) 44
GETCORNER (funzione) 44
GETDIST (funzione) 45
GETENV (funzione) 45
GETINT (funzione) 46
GETKWORD (funzione) 46
GETORIENT (funzione) 46
GETPOINT (funzione) 47
GETREAL (funzione) 48
GETSTRING (funzione) 48
GETVAR (funzione) 48
GRAPHSCR (funzione) 48
GRCLEAR (funzione) 91
GRDRAW (funzione) 91
GRREAD (funzione) 92
GRTEXT (funzione) 91
Gruppi di selezione in AutoCAD 88
Gruppi di selezione, funzioni 77

H

HANDENT (funzione) 82
Heap, memoria 95

I

I/O, funzioni
CLOSE 35
OPEN 59
PRIN1 60
PRINC 62
PRINT 62
PROMPT 62
READ 63
READ-CHAR 63
READ-LINE 64
TERPRI 69
WRITE-CHAR 74
IF (funzione) 49
INITGET (funzione) 49
Intere, costanti 4
INTERS (funzione) 51
Introduzione di pause, funzione
GETANGLE 44
GETDIST 45, 46
GETORIENT 46
GETPOINT 47
GETREAL 48
GETSTRING 48
ITOA (funzione) 52

L

LAMBDA (funzione) 52
Lanciamento dei programmi AutoLISP 104
LAST (funzione) 52
LENGHT (funzione) 53
LEXPLODE comando 116
Librerie di funzioni 39
Linea, tipi di (accesso alla tavola dei
simboli) 88
LISPHEAP variabile d'ambiente 95
LISPMEM variabile d'ambiente 95
LISPSTACK variabile d'ambiente 95
LIST (funzione) 53
LISTP (funzione) 53
LOAD (funzione) 39, 53
LOG (funzione) 54
LOGAND (funzione) 54
Logiche, funzioni
~ 28
AND 29
BOOLE 32
LOGAND 54
LOGIOR 54
OR 60
LOGIOR (funzione) 54
LSH (funzione) 55

M

Maiuscolo 4
MAPCAR (funzione) 55
MAX (funzione) 56
MEM (funzione) 101
MEMBER (funzione) 56
Memoria, gestione 95
Memoria minima 7
Memoria nodale (recupero) 97
Memoria virtuale 98
MENUCMD (funzione) 56
Messa a punto, funzioni di
TRACE 70
UNTRACE 73
MIN (funzione) 57
Minuscolo 4
MINUSP (funzione) 57

N

Nodale, memoria 95
Nomi di entità in AutoCAD 88
Nomi di simboli 4
NOT (funzione) 57
NTH (funzione) 58
NULL (funzione) 58
NUMBERP (funzione) 58
Numeri interi 2
Numeri reali 2

O

OPEN (funzione) 59
Operazioni di riordino 99
Opzioni 46, 50
OR (funzione) 60
OSNAP (funzione) 60

P

Paginazione virtuale di funzioni 98, 101
Parole chiave 46, 50
Pausa per immissione dati da parte
dell'utente 36
PAUSA, simbolo 36
PI (costante) 60
Piano (accesso alla tavola dei simboli) 88
PIRAMIDE comando 108
POLAR (funzione) 60
Polilinee, gestione tramite AutoLISP 88
PRIN1 (funzione) 60
PRINC (funzione) 62
PRINT (funzione) 62
PROGN (funzione) 62
Programmi applicativi 104
Programmi di AutoLISP 102
Programmi illustrativi 117
PROJECT routine 122
PROMPT (funzione) 62

Q

QUOTE (funzione) 5, 62

R

READ (funzione) 63
READ-CHAR (funzione) 63
READ-LINE (funzione) 64
Reali, costanti 4
REDRAW (funzione) 64
REF comando 116
Relazione, funzioni di
< 27
<= 27
> 27
>= 28
EQ 41
EQUAL 41
REM (funzione) 65
REPEAT (funzione) 65
RETE comando 108
REVERSE (funzione) 65
REVINFO disegno modello 118
RPOLY routine 122
RTOS (funzione) 65

S

S::STARTUP funzione speciale 39, 40, 53
S::XXX funzioni esecuzione automatica 40
SCATOLA comando 106
SCODELLA comando 107
Segmentazione della memoria 99
SET (funzione) 66
SETQ (funzione) 66
SETUP comando 116
SETVAR (funzione) 67
SFERA comando 109
SIN (funzione) 67
SLOT routine 122
Spazio di stringa 99
Spazio nodale 99
SPIRAL routine 123
Spline 88
SPLINESEGS variabile di sistema 88
SQR routine 123
SQRT (funzione) 68
SSADD (funzione) 80
SSDEL (funzione) 80
SSGET (filtri) 78
SSGET (funzione) 77
SSLENGHT (funzione) 79
SSMEMB (funzione) 81
SSNAME (funzione) 79
SSX ruotine 117
Stack, memoria 95
Statistiche di memoria 101
Stili di testo (accesso alla tavola dei simboli)
88
STRCASE (funzione) 68
STRCAT (funzione) 68
Stringa, funzioni di
STRCASE 68
STRCAT 68
STRLEN 68

SUBSTR 69
Stringhe 2, 4, 61
STRLEN (funzione) 68
Submenù MENUCMD 56
SUBST (funzione) 31, 68
SUBSTR (funzione) 69

T

TABLES routine 123
Tavole dei simboli, accesso 88
TBLNEXT (funzione) 88
TBLSEARCH (funzione) 90
TERPRI (funzione) 69
TEXTSCR (funzione) 69
Tipi di dati 2
TORO comando 110
TRACE (funzione) 70
TRANS (funzione) 70
Trigonometriche, funzioni
COS 38
TYPE (funzione) 72

U

UCS Sistema di coordinate di AutoCAD 70, 85
UNTRACE (funzione) 73
Valutazione, programma di 3
Variabili di sistema
AUNITS 29
AUSPEC 29
CMDECHO 35
modifica 67

V

Varie, funzioni
ERROR 75
LAMBDA 52
VER 73
VMON 98
VER (funzione) 73
Vista contrassegnata da nome (accesso alla tavola dei simboli) 88
VMON (funzione) 98
VPORTS (funzione) 73

W

WCS Sistema di coordinate di AutoCAD 70, 85
WHILE (funzione) 74
WRITE-CHAR (funzione) 74
WRITE-LINE (funzione) 75

Z

ZEROP (funzione) 75